VEHICULAR RACING SIMULATION: A MEL SCRIPTING APPROACH

A Thesis

by

HOBART CHAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2004

Major Subject: Visualization Sciences

VEHICULAR RACING SIMULATION: A MEL SCRIPTING APPROACH

A Thesis

by

HOBART CHAN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Frederic I. Parke (Chair of Committee)	Donald H. House (Member)
John Bryant (Member)	Phillip Tabb (Head of Department)

August 2004

Major Subject: Visualization Sciences

ABSTRACT

Vehicular Racing Simulation:

A MEL Scripting Approach. (August 2004)

Hobart Chan, B.E.D., Texas A&M University

Chair of Advisory Committee: Dr. Frederic I. Parke

The purpose of this thesis is to develop an automated technique for controlling the animation of computer-generated cars for the application of motorsports, also known as car racing. The basic idea is similar to previous work simulating flocks of birds and schools of fish. This simulation system provides a behavior model for each car and driver in a group of cars that enables them to race on a track while avoiding collisions. The technique is implemented using a commercial software package, called *MAYA*, utilizing its scripting language and built-in dynamics engine. While not a complete real-world dynamic simulation, the cars exhibit realism in both racing behavior and in visual motion attributes. This system allows the animator to control the number of vehicles, their properties, and their general path using an interactive interface. The automated technique replaces manual animation of each individual car and expedites production for animation or live-action effects film that includes computer-generated racing cars.

To my family and friends

ACKNOWLEDGMENTS

I would like to express my thanks to my thesis committee chair, Dr. Frederic I. Parke, for his guidance and for his attention to details. I would also like to thank my thesis committee member Dr. Donald H. House for providing additional knowledge, which this thesis uses, and to Dr. John Bryant for his show of enthusiasm throughout this process.

I also thank Jeffrey Smith and Lori Green for providing me direction and advice on the process of doing a thesis.

My gratitude goes to all of the faculty and staff of the Viz Lab who have enhanced my wealth of knowledge and have helped me reach my goals. Thanks to all the friends that I have made during my time in the Viz Lab, for their inspiration, comradery, and enthusiasm. Finally, I would like to thank Mom and Dad for their unconditional support and love through the years.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER 1

INTRODUCTION

I.1. Introduction

The simulation of groups or crowds of characters is widespread in animation and live-action productions featuring visual effects. Automating the motion control of birds, fish, and biped characters has been popular in computer animation. In most productions, such characters are only required to exhibit a generalized behavior. A classic example involves nothing more than a group traveling from point A to point B. This enables animators to repeatedly use the same technique with other types of characters for other productions. A herd of buffalo would travel in a much similar way as a herd of horses. Both animal groups have basic simulation properties such as avoiding collision, each character traveling at a velocity similar to the character next to it, and not going astray from the pack [11].

Extensive studies have been made to emulate real animal behavior. Rules of thumb have evolved that govern basic movement of herds, packs or flocks, whether with realistic or stylized behavior. This approach proves efficient when animating hundreds of thousands of characters (no matter what type they are) in a scene. However, in the more specific genre of group simulation, motorsports, the automated motion control of

---

This thesis follows the style and format of *IEEE Transactions on Visualization and Computer Graphics.*

vehicular racing has not been as well explored or implemented for computer animation. The reason being that vehicular racing behavior is less generalized and more dependent on context such as car commercial production or movies about car racing.

The responsibility of the animator is to give believable motion to a computer-generated character. The same principle applies to the animation of automobiles. If the scene involves more than one car, manually animating each car's movement and their interactions is tedious. The many possible combinations of manually created paths for the cars could look unnatural and are prone to error such as collisions [3]. Suppose the animator manually animates two interacting cars in a scene and later decides that more cars are necessary. The animator would have to recreate the movement of the first two cars to accommodate more cars for a natural looking behavior. Reanimating each car's path is time consuming. Rather than controlling every car's individual motion and path, the animator should just be able to control the car's behavior. Based on rules and defined behavior, the motion of each car is then automated in response to its environment. This approach is needed for efficient, robust, and believable vehicular racing simulation.

Assuming that a group of cars is just the sum of all the interacting behaviors of the individual cars, only one parameterized car behavioral model is needed. Simulating a group of cars means creating many instances of the simulated car model. During simulation, the individual cars move based on their parameter variables, local 'perception' of the virtual environment, and their driving dynamics capabilities. The aggregate behavior of the cars (no matter how many or how few) will look natural, much like in a real racing sequence.

To date, only the modeling and rendering aspects of computer-generated automobiles have benefited from increases in computing power. As the development of computing power advances and behavior simulation improves [9], simulated car racing results can appear more realistic, closer to the real world.

This research shares much in common with racing games. This work can be used to improve them. The implementation of the simulation techniques works much like a video game but without the requirement of constant user input and without the requirement for real-time performance. Compared to video games, the displayed results will be more compelling, having the flexibility of different rendering styles ranging from cel animation to photo realism.

CHAPTER II

RELATED WORK

Simulated behavioral systems of natural creatures have been explored in computer animation over the last two decades. Most of the research serves the purposes of entertainment, specific to a certain context, and is not true artificial intelligence. Artificial intelligence, from an academic standpoint pursues extraordinarily difficult problems such as modeling human and/or animal cognition [10]. The term "intelligence" is so ambiguous and misapprehended that the use of the term is avoided. Better terms would be "agent design" or "behavioral modeling." In the field of computer animation, agents are designed to produce an appropriate behavior for a particular application. The adaptability of true animal or human-like intelligence is not always necessary [10]. The ideas in this thesis stem from various past disciplines and research. Each had a different goal ranging from attaining more realistic simulation, to user-friendly out-of-the-box software packages, to efficiency in pipeline production.

II.1 Behavioral Models

II.1.1. Eurythmy

In 1985 Susan Amkraut, Micheal Girard, and George Karl from the Computer Graphics Research Group of Ohio State University produced an animation titled *Eurythmy* [1]. This animation featured a flock of computer-generated birds that fly out of a minaret through a series of columns, and down into a lazy spiral around a courtyard. Each bird avoids collision with its flock mates. The software used, informally called "the force field animation system" relies on two kinds of forces. There are "rejection" or repulsing forces surrounding each bird and each static object. Each bird also has a motivating force that varies as a function of its position. A transformation operator is used to calculate an acceleration vector based on the bird's current location (within predefined spatial fields).

II.1.2. Flocks

In 1987, Craig Reynolds developed the idea of *flocking* [11]. Reynolds' approach to *flocking* enables the birds to have a more natural behavior during flight. Rather than using repelling force fields, Reynolds uses "a distributed behavioral model," similar to a natural flock. In this technique, autonomous computer-generated birds determine their own path of flight based on the laws of physics and rules based on the action of other birds and the environment. The rules include collision avoidance, velocity matching, and flock centering. Collision avoidance is defined as individuals not colliding with other individuals. Velocity matching is individuals moving with a velocity similar to its neighbors. Flock centering occurs when an individual has gone astray from the flock and attempts to head back towards the center of the flock. The basic behavioral principles that Reynolds defined have been used by many of the successive flock systems in both gaming and computer animation.

II.1.3. Path Following

Craig Reynolds also came up with a system of path following where a character is steered along a predefined path such as roadway, corridor or tunnel [13]. This is different from constraining a vehicle to rigidly following a path, like a train rolling along a track. Path following is intended to produce motion much like humans walking down a corridor. This system allows the character freedom to deviate from the path, loosely following the general intended direction of locomotion. The implementation is based on the character following a goal that traverses along a pre-defined path centerline. This method provides a basic foundation for how characters can move along a path. However, development is needed to produce a more specific type of behavior suitable to ones own project.

II.2. Software Packages

A few behavior simulation systems have been written as stand alone software or plug-ins to accommodate the demanding needs of commercial animation production. Several of these are described below. These allow more time and resources to be spent on other tasks rather than in producing new behavioral engines.

II.2.1. Massive

Massive, a software system developed by WETA [6] can accommodate most crowd simulations. *Massive* was used in production of all three films in the "Lord of the Rings" trilogy. Although this software is mainly used for biped characters, it can be used for other types of group simulations as well. The interface design is friendly enough that artists can script character behaviors. A major disadvantage of this software is the lofty price for a license.

II.2.2. ArchVision RPC Automobiles

ArchVision RPC Automobiles is a plug-in for 3D Studio Max that provides a variety of cars as entourage for architectural renderings [4, 14]. The designer selects one of the many production cars out of a library to insert into a scene. The animation requires drawing a spline path for the car to follow and specifying its speed at desired locations on the spline. This simple plug-in, without any dynamics, is designed to produce still images and simple fly-through animations for client presentation purposes.

Rpc Automobiles cannot handle group simulations. The animator must specify each car's exact movement. *RPC Automobiles* arrive as a package with fixed options. The program structure behind the graphical interface is difficult to change. Only the basic input parameters can be modified.

Other software such as *SoftImage|Behavior* [7] enable easier animation of crowd systems, but may not be as easy to tailor to specific project requirements. Scripting and set-up are required. There is no known system designed specifically for animation of vehicular racing.

CHAPTER III

METHODOLOGY

III.1. Introduction to the Behavior Model

Traditional computer animation requires the animator to fully describe the character's motions in order to convey the desired characteristics to the viewer. This is inefficient when directing a group of characters that do similar, yet individual motions. Rather than explicitly describing each and every character's nuances, the motion details should be automated.

The approach for this project is to create a system that allows the animator to direct the motion of a group of cars around a racetrack without meticulously specifying the motion details. The system described simulates the aspects and attributes of car racing. This includes racing strategies, racecar dynamics, and behaviors of race drivers. There is a relationship that exists between a car's performance capability and a driver's skill. Winners are determined not only by how well a car performs but how good the driver is [5]. No two cars or drivers are exactly the same in terms of skill, behavior and physical dynamics. For purposes of this simulation, driver and car is treated as one entity, and are referred to as "autonomous cars" [12]. In the following we define the aspects and attributes of car racing.

III.1.1. Circuit Racing

The intent of this research is to simulate not just any kind of racing but circuit racing in particular. Unlike oval tracks, racing circuits (see Figure 1) are designed to include bends of all kinds as they are found on public roads [5]. This provides a considerable challenge that tests the performance capability of all aspects of a car and the driver's skill. Contenders range from amateur to professional drivers, and equally diverse cars. Daily commute cars, family sedans, and exotic sports cars are commonly found in this genre of motorsports. Some cars are heavily modified to maximize performance, while some cars are kept stock. Naturally, there are different divisions or classes by which the cars are ranked depending on their performance capability.

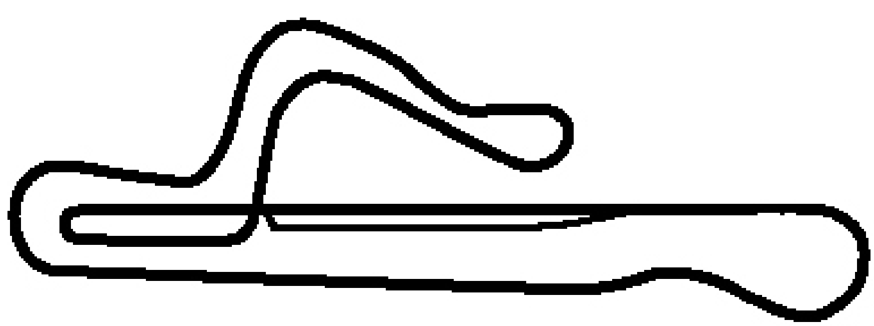


Fig. 1. Map of the Central Circuit in Hyogo, Japan.

III.1.2. Knowing the Track

There is a considerable difference between racing on a course that is not well known by the driver (such as rally racing) and racing on a track where the driver knows every bend and turn through a succession of practice laps. We focus our attention on circuit racing where drivers know the track prior to a race. In addition, racing circuits (see Figure 2) are paved and relatively flat, unlike off-road desert and rally racing (see Figure 3). Track conditions are more predictable without the constant changes typical on dirt or gravel courses. Through practice runs, drivers familiarize themselves with the many details and subtleties of a track developing an almost photographic image of it in their minds [2, 5]. Drivers progressively learn the limits of their driving ability as well as the performance limits of the car imposed by natural forces during braking and cornering. On race day drivers know exactly where to turn, when to brake, and when to accelerate. There is no room for improvisation. Knowing most of the factors of the problems involved beforehand, the approach is almost scientific, allowing a driver to repeat the same performance lap after lap.

(A)

(B)

Fig. 2. Examples of racing circuits: (A) SCCA World Challenge, 2003, image used with permission of Jonathan Heiliger, Motorsport.com. (B) SCCA World Challenge, 2003, image used with permission of Jonathan Heiliger, Motorsport.com.

(A)

(B)

Fig. 3. Examples of world rally championship racing: (A) Argentina, 2002. Image used with permission of Subaru World Rally Team. (B) Acropolis Rally, 2003. Imaged used with permission of Subaru World Rally Team.

III.1.3. Vision

An intelligent racer must have the ability to constantly look ahead at the road conditions. This allows time to anticipate what will happen, as well as reacting to events that are currently occurring [10]. Timing is crucial. For example, a driver knows that a reduction in speed is necessary before entering a sharp bend. Braking too early results in loss of speed, while braking too late may cause a loss of control. The driver must find and maintain the ideal speed allowable, at the limit of tire grip, to take the bend safely. Failure to plan ahead or a slight miscalculation can result in a loss of precious seconds, the car sliding off the track, or even accidents.

III.1.4. Look-Ahead Distance

The look-ahead distance needed is largely determined by track conditions and the car's speed. Drivers will go faster on straightaways, on wider and flatter roads, and on roads with no obstacles. They must look further ahead as the pace of events becomes quicker (see Figure 4(A)). This gives extra time to react to upcoming events. Conversely, when driving down a narrow path, taking a curve, or on a road that has obstacles, drivers need to slow down to sample more information (in a more local region) in a given time frame [10]. This allows more precision on steering correction and overall balance (see Figure 4(B)).

When it comes to this look-ahead distance, there are no set values. Due to the many variables, the distance will vary in each situation. For example, the look-ahead distance may not be as far on a wide track if there are cars in every lane. In a perfect scenario where the road is very wide with no curves or any kind of obstacle, the look-ahead distance at the horizon would be the ideal.

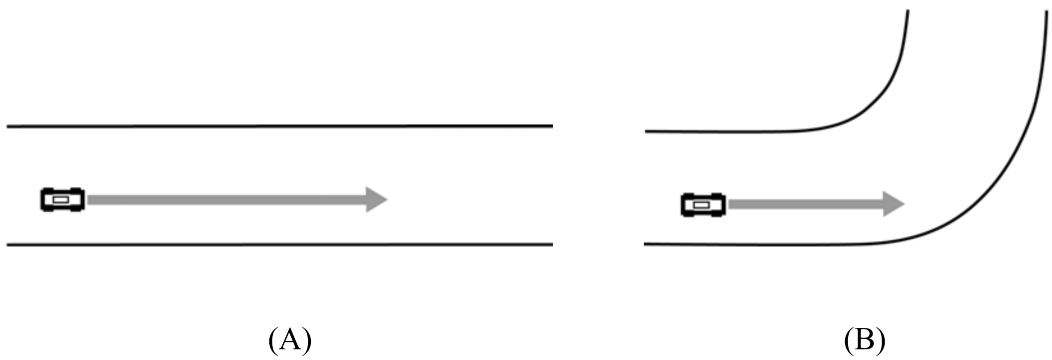


Fig. 4. Look-ahead distance: (A) moving fast on a straightaway, (B) moving slowly before a bend.

III.1.5. Setting Targets

During the race, the driver makes a set of visual targets or aim points at incremental distances along the track (see Figure 5). Distance to the next target depends on the visual information the driver receives (road condition, speed of car, obstacles). Once the driver successfully reaches the current target, the driver looks further ahead to set another target. This is seamless and natural for a driver without him ever being conscious of it.

To simulate this, a general road path for the group of cars to travel is specified [13]. Each car will have its target to follow. The target advances along the path as the car approaches its current target location [13]. The faster the car goes, the faster the target advances.

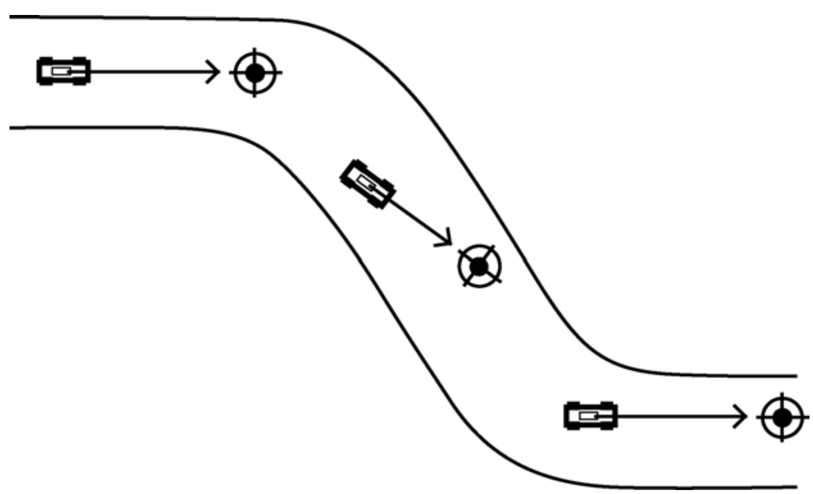


Fig. 5. Setting targets at incremental distances.

III.1.6. Boundary Confinement

Of the many rules and regulations of racing, staying within the boundaries of a track is mandatory. This controls the flow of traffic, keeping contenders from taking shortcuts and providing a safe distance from the spectators. Failure to comply may result in slower lap times, accidents, or even disqualification. Drivers rely on their look-ahead distance to judge the left and right boundaries of a track, and to maintain their position between the boundaries. They will drive more to the left or right side of the track depending on what lies ahead.

III.1.7. Acceleration

During a race, a driver will accelerate under full throttle (most often on straightaways) unless an external factor prohibits it. Factors include bends, obstacles, and substance on the track that reduces traction (snow, gravel, and bits of shaved rubber from other tires). Acceleration is defined as the rate of change of velocity with respect to time. Modern production vehicles can accelerate from a standstill to 60 miles per hour in anywhere from 3.5 seconds (exotic sports cars) to 15 seconds (pickup trucks). The acceleration varies from car to car and depends on power output from the motor (torque and horsepower), load (weight of the car and coefficient of drag), and how well the driver delivers that useable power to the pavement (efficient up-shifts and minimizing tire slippage) [5]. Acceleration is decreased going up a slope and the opposite when going down because of gravity. Amateur and purpose-built racecars alike are modified and/or built to maximize power-to-load ratio. For example, a lighter car will accelerate quicker than a heavier car having the same amount of power. Likewise, both amateur and professional drivers will master their driving skills to make the most out of their car's performance capability.

III.1.8. Braking

Brakes are applied when a reduction in speed is necessary to go through a turn, or to avoid approaching obstacles. If the driver is not accelerating, the driver is braking. It is rare during competition that a driver is doing neither. Braking is the opposite of acceleration. It is also known as deceleration. Braking performance also varies depending on the mass of a car, brake rotor and caliper size, brake pad material composites, and rate of heat dissipation [2]. The braking system is more powerful than any other system in a car. Therefore, a car is capable of decelerating more quickly than it can accelerate, (even more so going up a hill and vice versa going down). Modern production cars can stop from 60 miles per hour in distances ranging from 100-150 feet, and even less on purpose-built racecars. For this reason, only a small percentage of time during a race is spent braking. Most drivers agree that to be competitive, brake usage should be minimized. Excessive braking can be costly, resulting in longer lap times and quicker brake wear. Care must be taken to brake smoothly instead of making abrupt stops. This eliminates wheel lock-up and loss of steering control.

III.1.9. Cornering

Racing circuits are different from other kinds of racing because of the many bends and turns involved. This makes circuit racing much more challenging and interesting when compared to straight-line drag racing or even oval racing, which requires only making wide unidirectional turns. Knowing how to effectively handle a bend or a succession of curves is the most important technique in circuit racing. Engine power is of little significance if either the driver or the car cannot handle turns [5]. Cornering ability is the biggest factor in separating the winner from the also-rans.

III.1.9.1. Grip

The principles of cornering are based on the natural forces acting upon a car as it changes direction. The ability of a car to move relies on the adhesion of its tires to the road. This is governed by the coefficient of adhesion, which varies with different road surfaces and tire design, and is directly proportional to the weight of a car [5].

$$A = W\mu$$

W represents the car's weight and μ is the coefficient of adhesion. Under optimal road conditions, μ is between 0.8 and 1.0 for road tires. For racing tires, μ can be as high as 1.5 [5]. For example, if $\mu = 0.8$, it will take a force of at least 2400 pounds to make a 3000 pound car slide sideways. If all four wheels were locked, it would take that same amount of force to move the car forward or in any other horizontal direction. As long as a car is moving at a constant velocity on a straight line, adhesion remains intact resisting lateral forces. Once the car is under braking, acceleration, or turning, additional forces are created which use up part of the adhesion [5].

A graphical way of representing adhesion can be the use of a traction circle graph (Figure 6). G is a unit of acceleration equal to the acceleration exerted by gravity (32 feet/sec^2). G can also be used to indicate the amount of lateral acceleration a car and tire combination can handle before the car begins to slide in any direction. One g-force is equivalent to the force of 1 times the weight of the vehicle [2]. If a 3,000 pound car is braking at 1g, there is a braking force of 3,000 pounds, likewise for both straight-line and lateral acceleration. Traction should always be used to its limits. Otherwise, unused available traction is wasted and the car will be slower than possible (see Figure 7(A) and 7(B)).

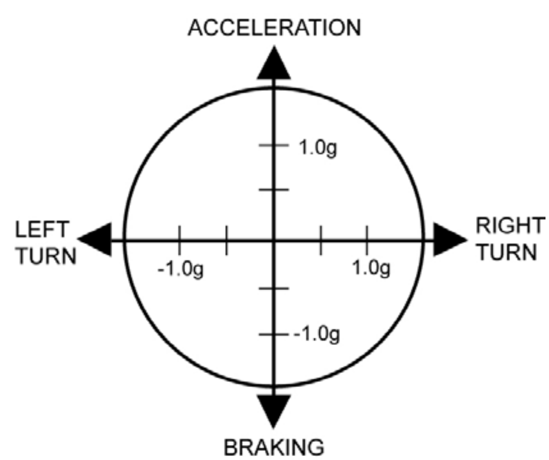
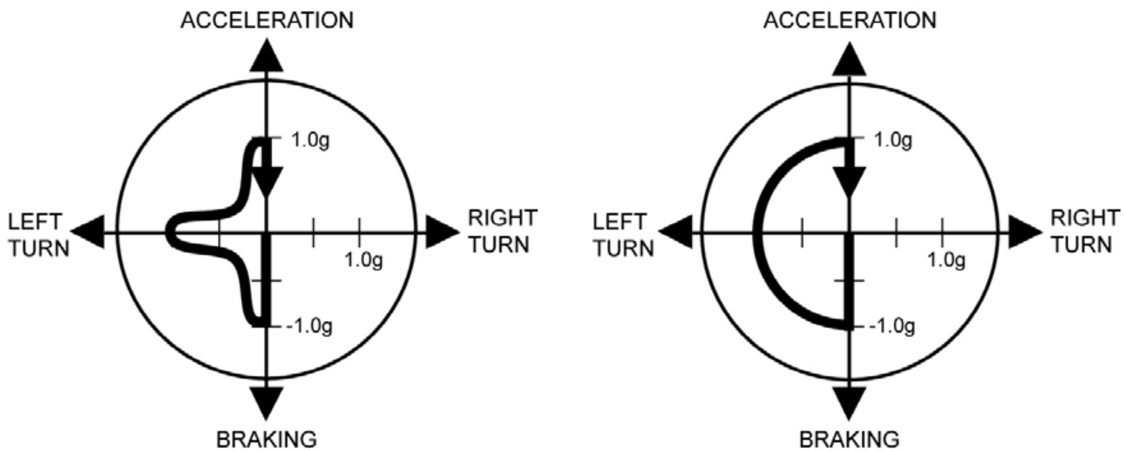


Fig. 6. Traction circle graph.



(A) (B)

Fig. 7. Traction circle graphs: (A) tire grip is not used to full potential, (B) tire grip is used to full potential.

In a situation involving a change from braking to turning to accelerating, at-the-limit g-forces must smoothly transition from one to the other. A car cannot instantly change direction of g forces. This means progressive blending of braking, cornering, and acceleration (see Figure 8). The ratio of each is a varying constant trade-off during a race and should always amount to a maximum of 100% of the traction limit. The driver wants to optimize performance by staying on the maximum traction circle. A sudden change of force in any direction will overload the tires, spinning the car out of control. For example, if 100% of the traction is used to corner, not even 1% of that traction can be used for braking [2]. If the traction to corner is reduced to 10%, 90% of the remaining traction can be now be used for braking.

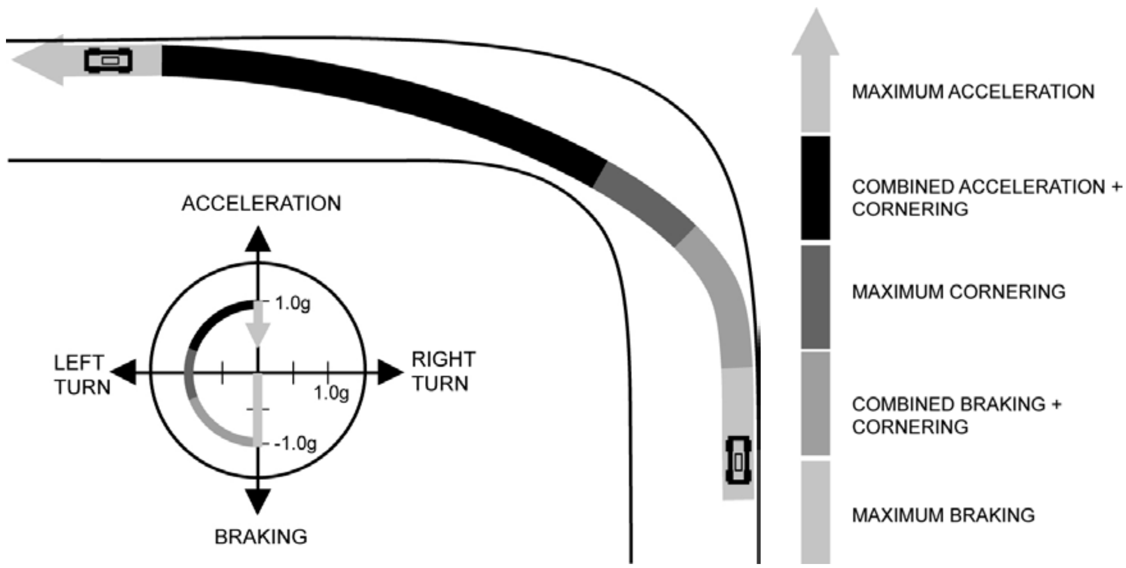


Fig. 8. Relationship between the driver's actions and the traction circle graph.

Under racing circumstances, if the car 's traction limit is not maximized by acceleration, it is either being used by braking or cornering. If turning occurs, a car is subjected to centrifugal force. This force can be represented as:

$$F_c = (mv^2)/r$$

where m is the mass of the car, v is the velocity, and r is the radius of the curve. The direction of the centrifugal force is along the axis from the center of the turning radius out to the center of the car's mass (where m = weight/gravity). Since weight is a factor in both adhesion and centrifugal force, a relationship exists between all three. More importantly, from the equation, centrifugal force is proportional to the square of the velocity and inversely proportional to the radius of the curve [2]. If the centrifugal force acting upon the car exceeds the limit of the car's adhesion, the car will slide. The driver will have to either slow down or turn in a larger radius to decrease the centrifugal force. Racecar drivers normally choose the latter, maximizing use of road width (with a larger turning radius) to corner at higher speeds while maintaining a margin of safety [5]. Thus, the greatest turning radius is inscribed in a given curved section of a road (see Figure 9). This turning path is known as the geometric line. Leading up to the turn-in point, the car is on the outer most side of the track and closes in on the inner most side barely hitting the apex of the corner, and then exits the turn heading back towards the outer most side of the track. This is assuming the car is at a constant cornering velocity. Such an approach is theoretically the fastest constant velocity path through a corner, but not always the fastest way to win a race.

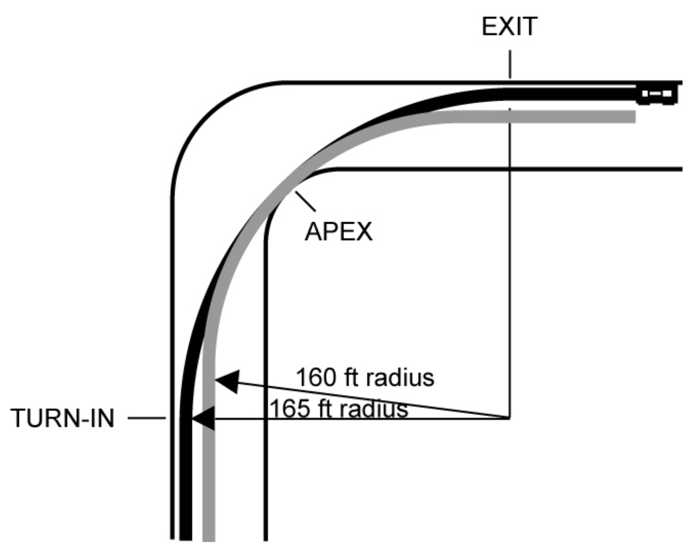


Fig. 9. Geometric line maximizes road width to reach higher cornering speeds.

III.1.9.2. Optimizing Available Grip

The aim of a racer is not to drive through every part of the track as fast as possible but to complete the current lap in the quickest in overall time [5]. This means driving through the corners in a way as to maximize straightaway speed. It is much easier to pass on straightaways than through corners. The focus for drivers is to modify the turning path from a constant turning radius to a variable turning radius. This results in the ideal line (see Figure 10). Typically, the ideal line has a sharper turning radius in the beginning and progressively straightens out to a larger radius towards the exit. Having to take a sharper curve in the beginning, the car cannot go through the bend as fast as before. However, this enables the car to dive deeper into the corner before braking, and to be straightened out quicker (upon corner exit) creating a longer straightaway than the geometric line would allow. Keep in mind that maximum acceleration can only occur on straightaways. The earlier a car is straightened out for the following straightaway, the earlier a driver can accelerate. This is a compromise of cornering speed for straightaway speed. The time to take a corner should be minimized to maximize the time spent on straightaways. Exit speed is far more important than cornering speed. The ideal line approach proves more effective than taking the geometric line.

While many will choose to take the ideal line, others will adapt variable degrees of both. This depends on driver skill, road conditions, and performance of the car. Cars with superior acceleration and braking ability will opt more for the ideal line approach whereas cars with superior cornering capability will opt more for the geometric line.

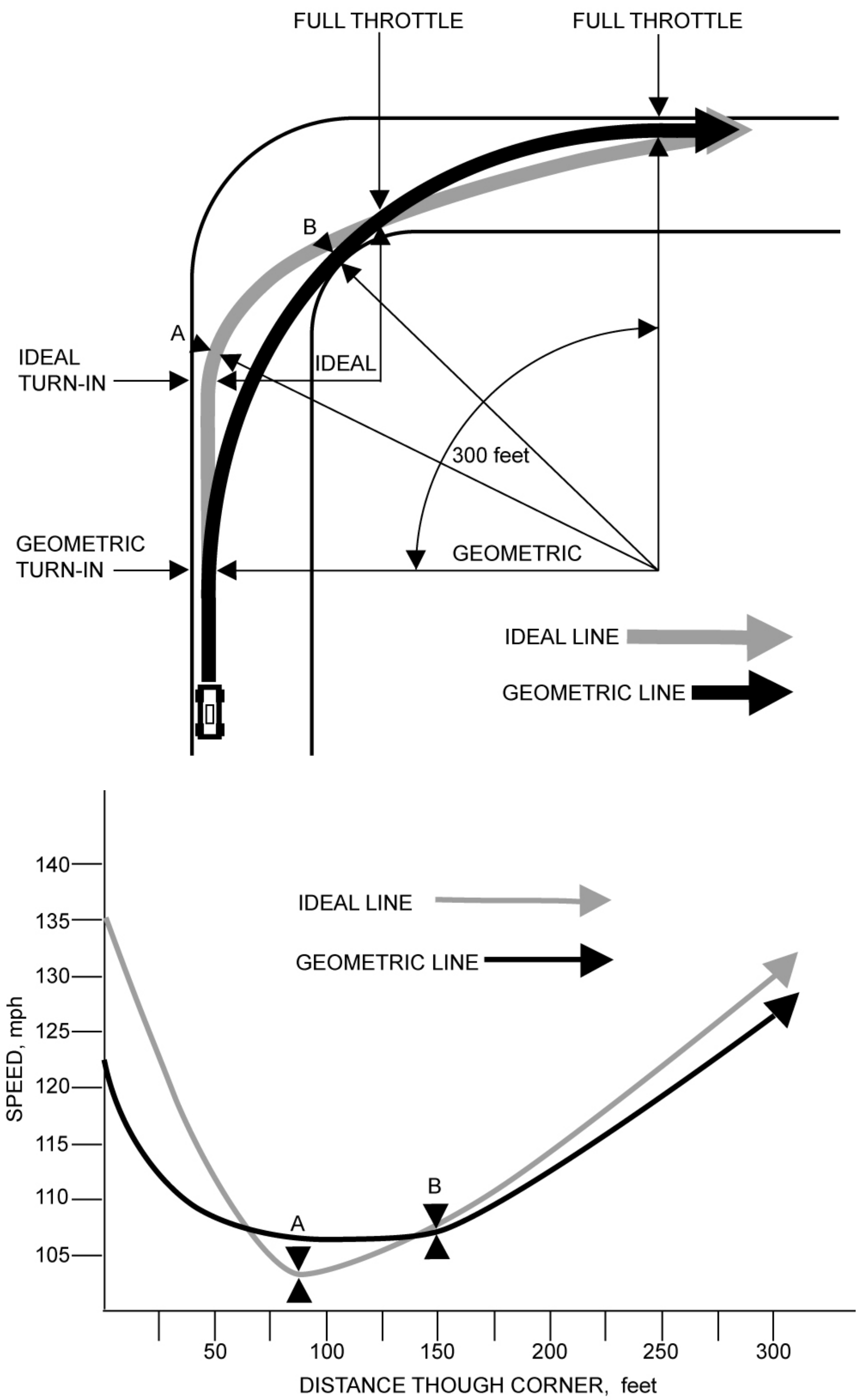


Fig. 10. Distance through corner versus speed [2].

To analyze this in more detail; there are six phases of control for the cornering technique; braking, trail braking, transition, balanced throttle, progressive throttle, and maximum acceleration [2] (Figure 11). These phases are a blend of throttle and braking. The duration and timing from one phase to the next depends on the type of corner, the driver, and the car. Trail braking is defined as turning and braking at the same time. The more a turn is progressively made, the more the brakes are released to keep within the traction limit. Transition is the phase between braking and the beginning use of the throttle. Balanced throttle maintains speed. Finally, when the car is straight enough, progressive throttle begins the acceleration phase and increases to maximum as the car exits the turn.

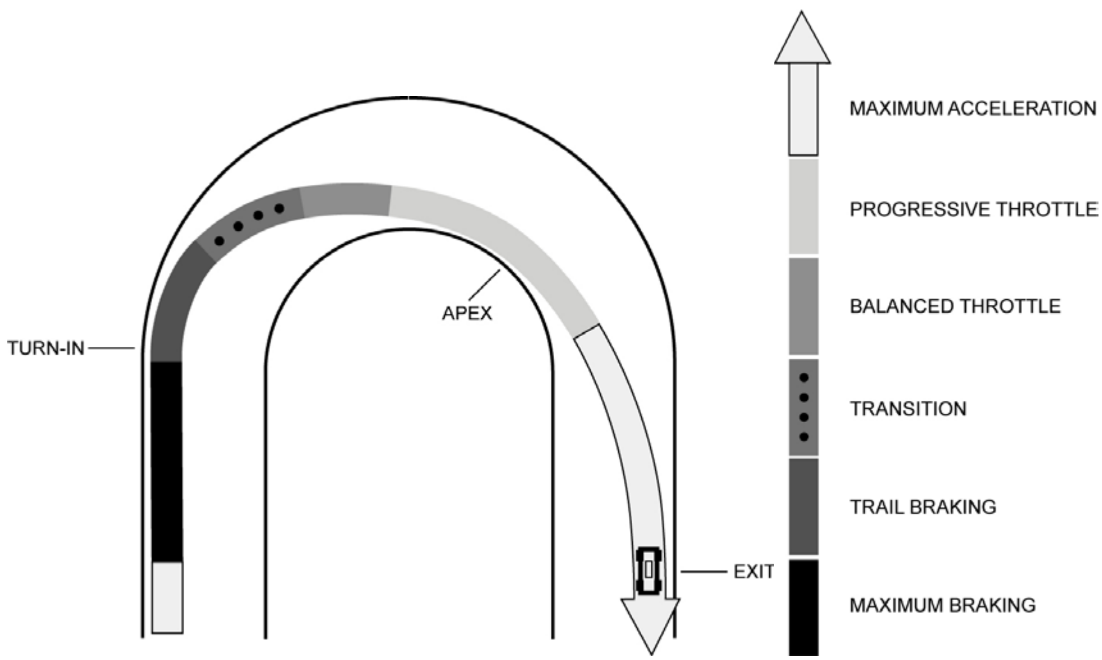


Fig. 11. Control phases through a 180 degree hairpin turn.

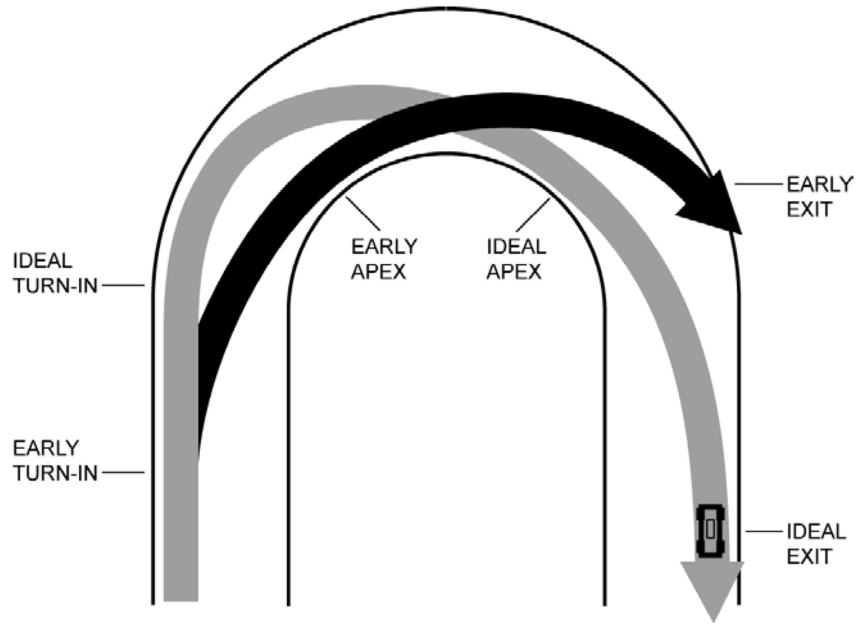
To execute the sequence of actions correctly, a driver will utilize three reference points to guide himself through a corner: turn-in, apex, and exit (see Figure 12(A) and Figure 12(B)).

Turn-in is where the initial turn takes place. This is the most important of the three reference points because this will dictate the speed of the car as well as where the other two reference points will be located. If the turn-in point is late, then both the apex and the exit point will come late in the turn as well. Cars that have superior braking ability utilize late turn-in points to maximize straightaway speeds [2].

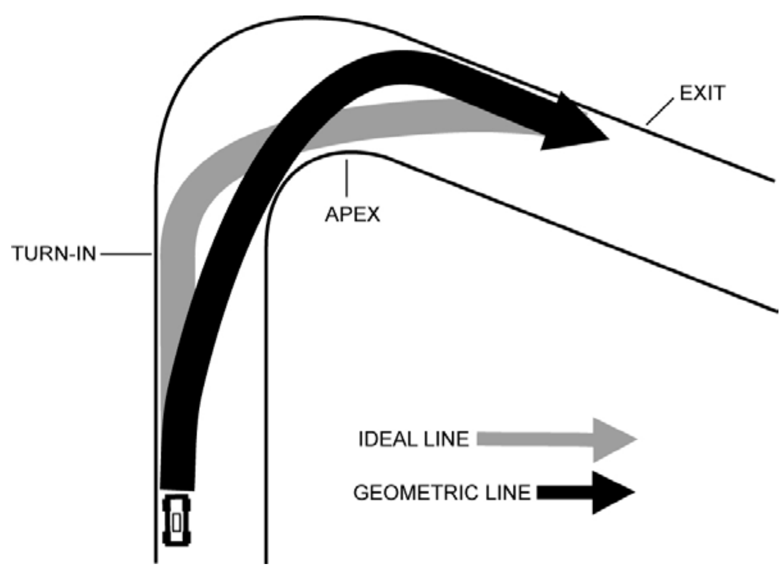
Apex is the point where the car is no longer driving in but driving out of the corner. This is where the inside wheels run closest to the inside of the curve. Location of the apex affects how the car exits [5].

Exit is the point where the direction of the car becomes tangent to the outer lane. The ideal exit point uses the full width of the track when coming out of the apex. If the turn becomes too wide, having to slow down or to turn the steering wheel tighter, then the apex was taken too early. If coming out of the exit leaves room to spare, then the apex was taken too late. When done perfectly, the car will stay barely within the boundaries of the track, accelerating as early and as hard as possible [2].

In rare situations a driver will utilize an early turn-in. This is either through inexperience or for emergency purposes such as regaining control prior to entering a curve. This allows more distance to brake in a straight line to regain control. One drawback of making an early exit is that the radius of the turn becomes progressively tighter (opposite of an ideal exit), which results in a slow exit speed.



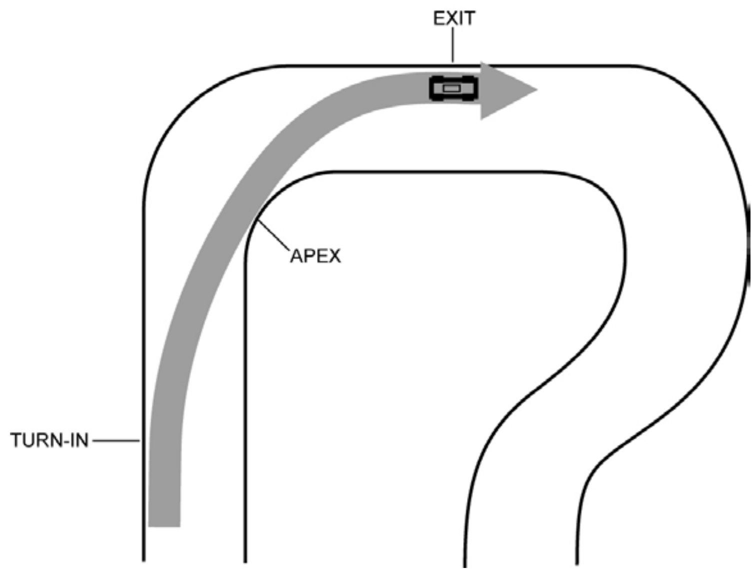
(A)



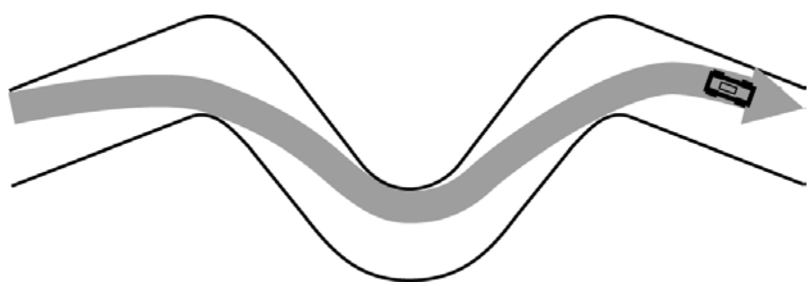
(B)

Fig. 12. Early versus late reference points: (A) 180 degree turn, (B) 120 degree turn.

The ideal line through a curve is not the same for all types of corners. For curves that do not end up connecting to long straightaways, an early turn-in approach is taken to maximize the entrance speed [2] (Figure 13 (A) and 13(B)).



(A)



(B)

Fig. 13. Curves that do not connect to long straightaways: (A) two consecutive 90 degree turns, (B) consecutive S-curves.

III.1.10. Passing

The ability to win races depends on knowing how to pass an opponent. Straightaways are the easiest place to overtake a car because the driving line is not drastically altered. The driver must first be aware of the surroundings and identify which cars are ahead. Cars that are behind (outside of the 180 degree field of view) are disregarded. Of the ones that are ahead, the driver must determine which one is closest. When the driver closes to within a minimum safe distance of the closest car ahead, the driver must either go left or go right to pass, or slow down and stay behind.

The driver must not miss out on opportunities to pass on corners as well,. Passing on a corner requires more skill because the ideal line must be altered to avoid collision. It is important that the driver passing must remain beside his opponent on the inner lane (by matching his braking with the opponent's) before the turn-in is taken (see Figure 14). This allows the passing driver to make himself/herself known to the other driver, and to block. Once the opponent identifies the passing driver, the opponent has no choice but to follow the passing car when making the apex leading to the exit. The opposite will happen if the passing driver does not make himself known by the time the turn-in is taken. The opponent will block the passing car; as the opponent is not aware of the presence of the passing car. Another incorrect way of passing an opponent is passing the opponent before making the turn-in (see Figure 15). The passing driver will then overshoot the turn, not be able to accelerate as early, giving the lead back to the opponent.

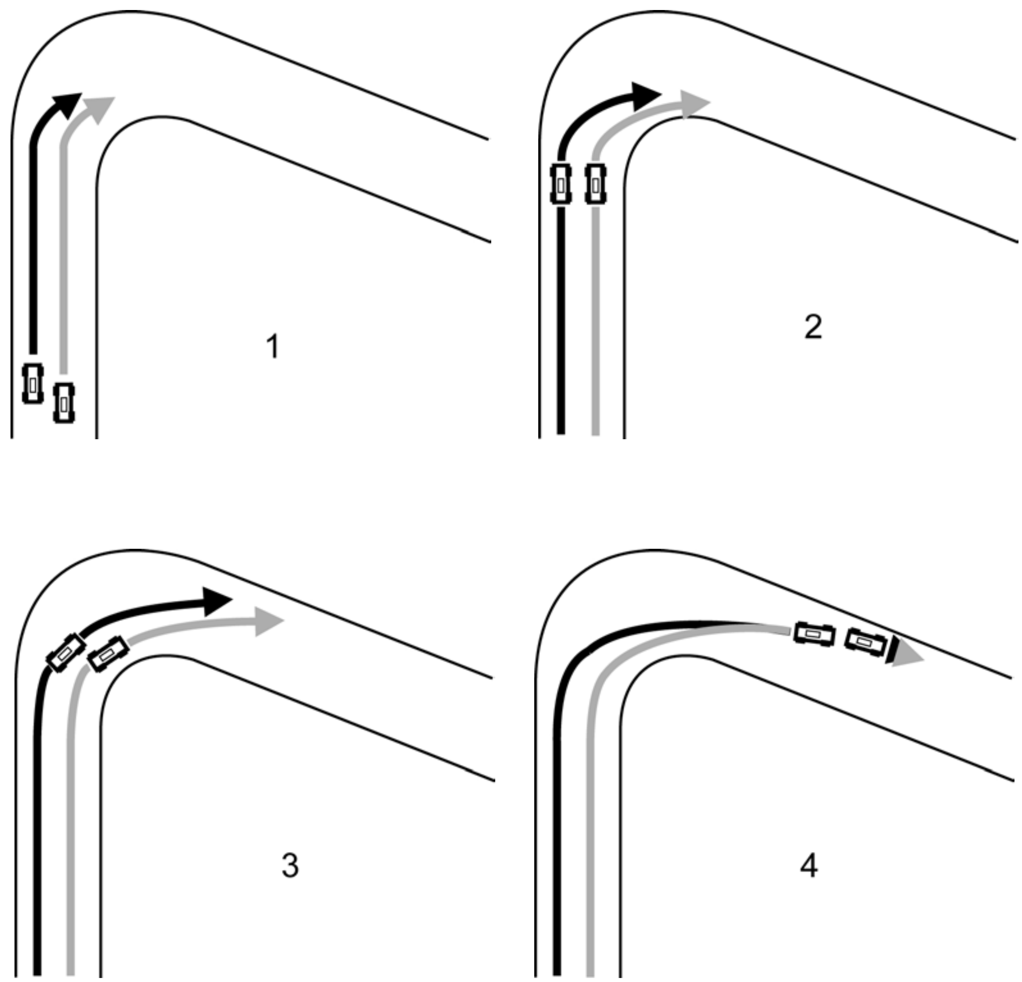


Fig. 14. The correct way to pass on a corner.

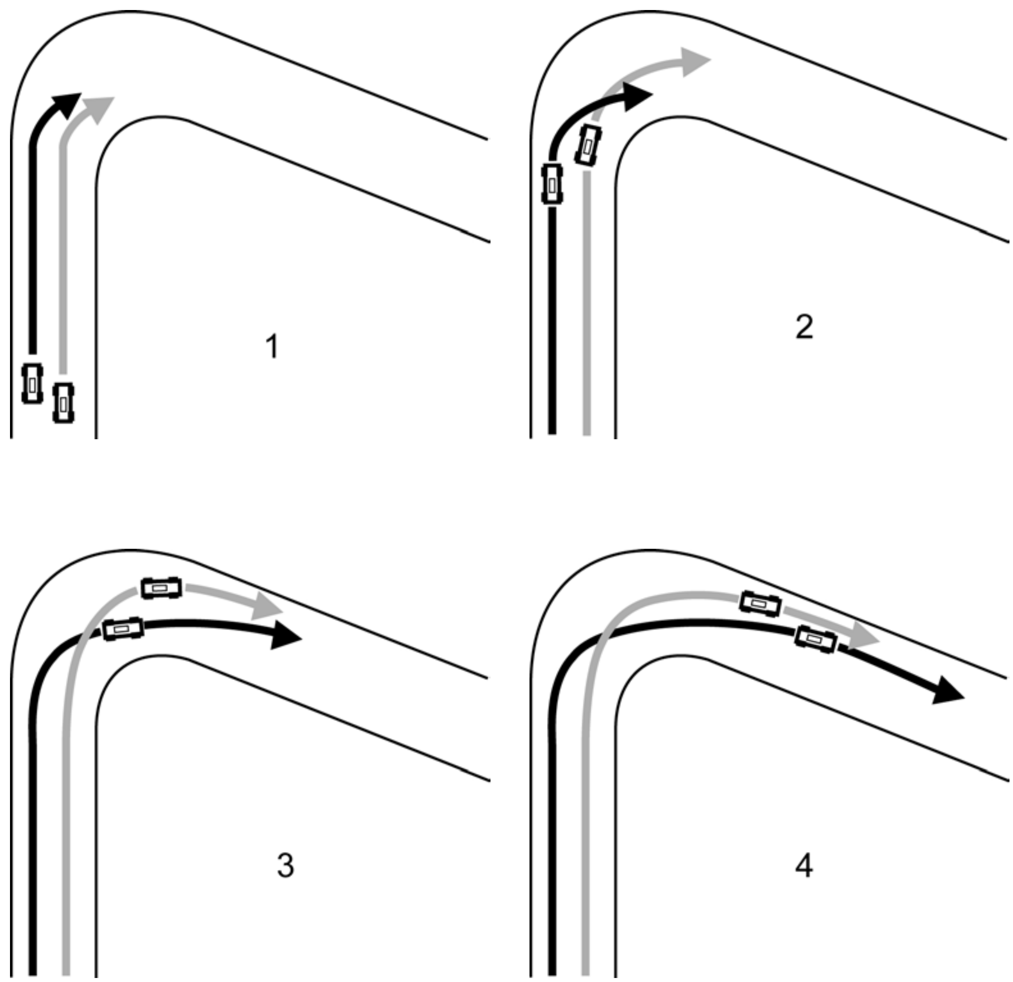


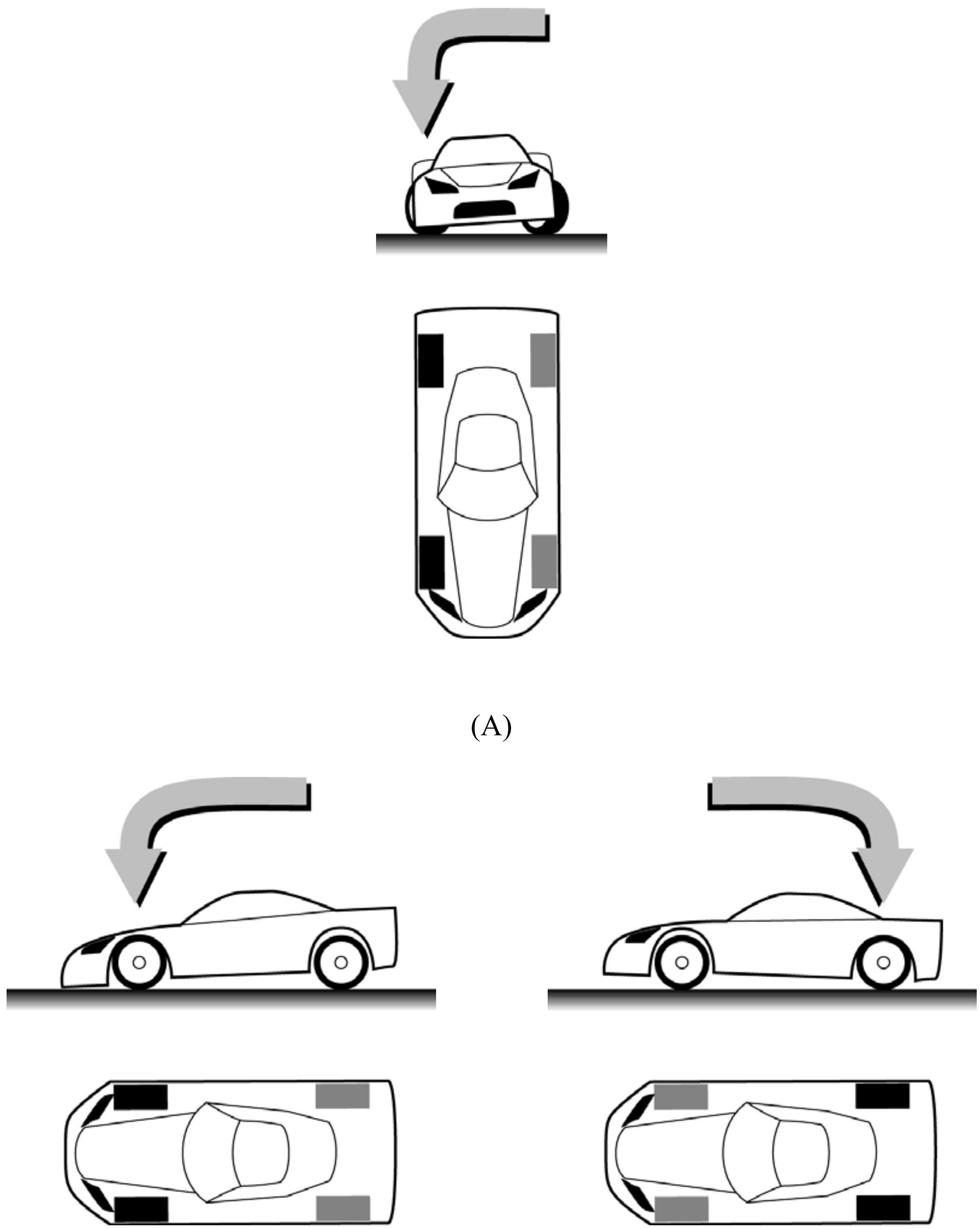
Fig. 15. The incorrect way to pass on a corner.

III.2. Suspension

Suspension systems, consisting of springs and shock absorbers, stabilize the body of a car when going over bumps and undulations. It also keeps tires in contact with the pavement at all times to maintain traction. In a perfect world where roads are glass-smooth, there would be no purpose for suspension systems.

A spring can be defined as an elastic coil that compresses or extends depending on the exerted forces. There is a spring at each corner of the car. Springs serve the purpose of supporting the body of the car. The purpose of a shock absorber is to dampen, or dissipate, the oscillations of a spring as it absorbs the undulations of a road surface. Shocks work in both directions: compression and extension. Without shock absorbers, springs would bounce at a much higher amplitude after hitting a bump and continue on for a lengthy period of time.

Weight transfer is inevitable when a car is taken to its traction limit. In the presence of lateral forces, the suspension system allows the center of gravity to shift. This directly affects spring compression as more weight is transferred to one or more corners of the car. This weight transfer causes body roll, nose dive, or rear end squat (see Figure 16).



(A)

(B) (C)

Fig. 16. Weight transfer: (A) body roll during cornering, (B) nose dive under braking, (C) back end squat under acceleration.

CHAPTER IV

IMPLEMENTATION

IV.1. Implementation Environment

The environment this simulation is developed in is a commercial 3D modeling, animation, and rendering package called *MAYA* [8]. One of the capabilities that *MAYA* offers is the option of writing procedures using *MEL* (Maya Embedded Language) scripts. This expands the users' ability to expedite specified animation tasks, and not be limited to strictly using the provided graphical interface [15].

Procedures written in a *MEL* script can quickly perform tasks without manual input. This is convenient when performing hundreds of calculations and/or repetitive tasks in a small amount of time. *MEL* scripts also enable creation of custom graphical interfaces such as pop up menus with buttons and sliders.

The autonomous motion and interaction between cars on the track are controlled by means of procedures written in a single *MEL* script. These procedures make some use of *MAYA*'s built-in functions and tools to create an approximate model of realistic car racing behavior. This simulation requires a pre-existing 3D model of a racetrack and at least one pre-existing 3D car model. Manual set up of the track and of each car is required for the script to distinguish these as distinct entities. Since the script recognizes objects by their names, we use a naming system to define the key objects of each car as well as the boundaries of the track. The simulation script is executed in the project file that has the track. The cars (existing in other files) are then imported onto the track via a graphical interface that allows control over the number of cars, the selection of car models, and the selection of different performance and behavior attributes for each individual car. This script is not restricted to using particular car or track models. The simulation is started after all attributes are specified.

IV.2. Simulating Behavior

While the approach is intended to create believable racing behavior, it is not necessary to exactly model real life behavior. Approximate representations are used throughout this simulation. These include simplified versions of the actual perception, thought process, and action of a real driver in a car race. We must first describe a practical representation of a racetrack for this simulation system. The basic track setup is the foundation upon which the simulation is run. We must then understand the implementation of how an autonomous driver perceives, thinks, and reacts to the conditions of a track as well as to the actions of his opponents.

IV.2.1. Defining the Track

The intent of this simulation is to model circuit racing where the driver knows the track before the race. It is appropriate that we define the racetrack with spline curves. The *outer curve* defines the rightmost boundary of the track while the *inner curve* defines the leftmost boundary of the track (see Figure 17). Although the cars will follow the

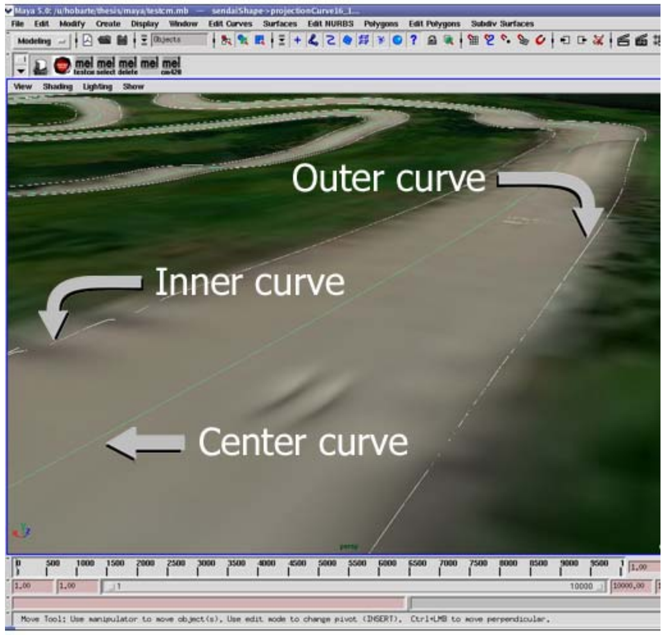


Fig. 17. Defining the racetrack with the outer curve and the inner curve.

predefined inner and outer curves, they will not rigidly follow the track like a train on rails, but merely use these curves as a guide. The *edit points* (or ep's) are evenly spaced point pairs located directly on both curves. Each pair of edit points (consisting of an edit point from the inner curve and an edit point from the outer curve) is identified with an index number, from 0 to the maximum number of points.

$$ep[n], ep[n+1], \ldots ep[max]$$

The direction in which these edit point index numbers increment determines the direction of car travel (see Figure 18). The e1 and e2 points are used to follow the track boundaries. These points move from one edit point to the next as needed. The e3 point is the average distance between these two. The target position moves left or right of e3 (perceived from the point of view of the car), for finer direction control. Each time the car comes within a minimum distance to the target (the visual aim point which the car follows) (see Figure 19), the edit point index increments allowing e1 and e2 to move to the next pair of edit point locations. This in turn moves e3, which moves the target. While the targets will not move outside the boundary limits, it is possible for the cars to drive outside of the boundaries.

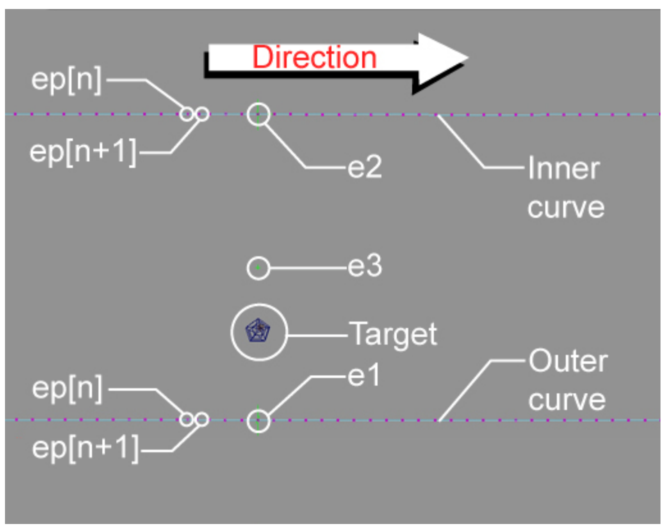


Fig. 18. Layout of the target.

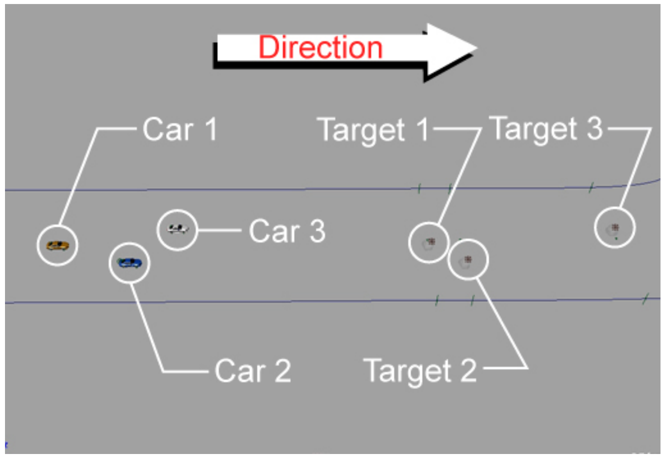


Fig. 19. Aerial view of cars to the respected targets.

IV.2.2 Autonomous Behavior

The goal is to produce autonomous behavior that emulates realistic racing behavior. Each driver evaluates the current conditions and acts on a time step by time step basis [16]. Each time step is 1/30th of a second, equivalent to one frame of animation.

IV.2.2.1. Look-Ahead Distance

It is crucial that the simulation looks ahead at the road conditions to allow time to anticipate future events as well as what is currently happening. It needs to know when to brake and turn before a curve and when to accelerate on a straightaway. The look-ahead distance is proportional to the velocity of the car, driver skill, and inversely proportional to the curvature of the road. The curvature of the track (measured in theta) is computed in the look-ahead region located ahead of the target (see Figure 20). The “c1”, “c2”, and "c3" points are the midpoint locations of "a1 and b1", "a2 and b2", and "a3 and b3", respectively. The points "a1", "a2", "a3" and "b1", "b2", "b3" move along the inner curve and outer curve edit points the same way as e1 and e2. Theta is computed using the vectors <c2, c1> and <c2, c3>.

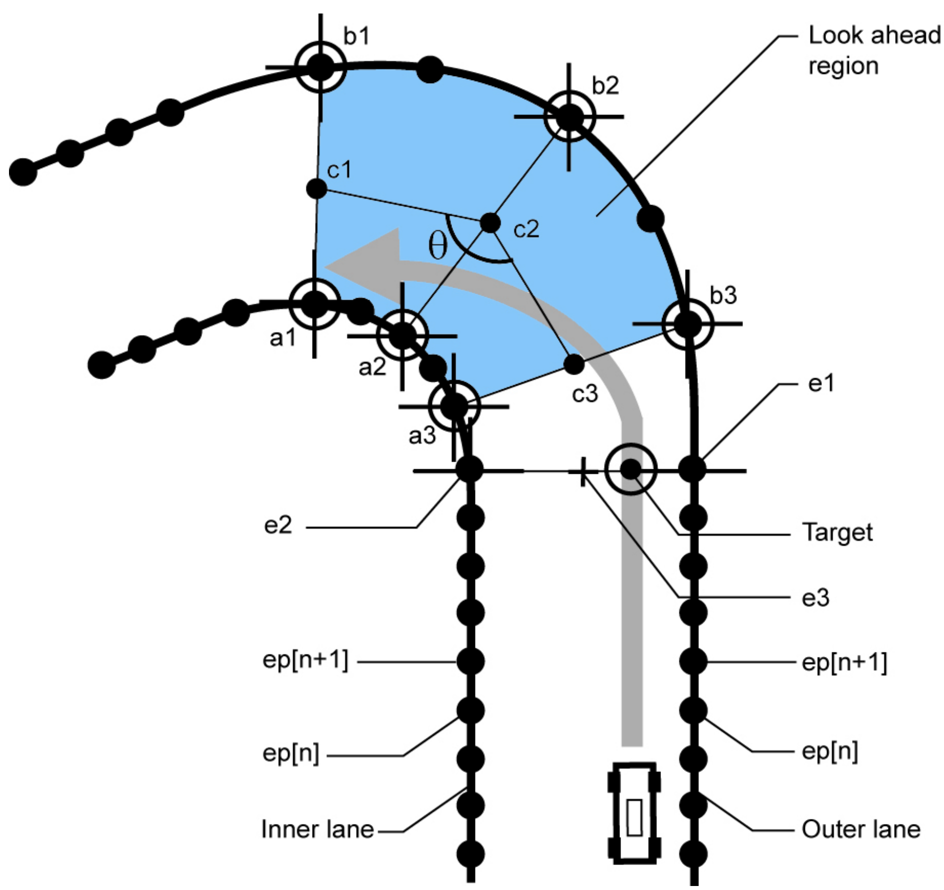


Fig. 20. The look ahead region computing the curvature of the track.

IV.2.2.2. Moving the Car

The method by which the cars follow their respective target is the use of *MAYA* dynamics "radial fields". Each target possesses a radial field that attracts its respective car. The target is like a string or a magnet that pulls the car around the track. The strength of the attraction is proportional to the magnitude of the value, which varies depending on road conditions.

IV.2.2.3. Acceleration and Braking

The change in magnitude of the attraction controls the acceleration and braking of a car. Acceleration and braking are rates of change in velocity with respect to time. If the attraction strength of the target decreases to a smaller value over time, the result is deceleration. Acceleration occurs if the attraction strength of the target increases over time. When a car accelerates from a standstill, the attraction value is initialized at zero and increments till it reaches the limit. This is the point where the car has reached top speed. Top speed is proportional to the allowable maximum attraction value. Acceleration and braking ability is proportional to the size of the increment or decrement of this attraction value per time step. The larger the decrements per time step, the greater the acceleration capability, and vice versa.

IV.2.2.4. Cornering and Steering

Maximizing road width usage during cornering is implemented by having the car loosely follow the target. This produces a lag of where the car ends up compared to where it wants to be.

For curves that connect to long straightaways, the ideal line is implemented. The look-ahead region computes the curvature of the track and the turning direction. The target position moves in the direction opposite of the turn creating a larger turning radius before the turn-in point (see Figure 21). For example, if the track curves to the right, the target will progressively move to the left relative to the car. This creates a late apex enabling the car to straighten out earlier for the exit. On curved roads with short straightaways, the quick change in direction allows little time for the target to progressively move left or right. As a result, the target stays relatively in the center. This creates an appearance that the car is taking an early to mid apex (see Figure 22).

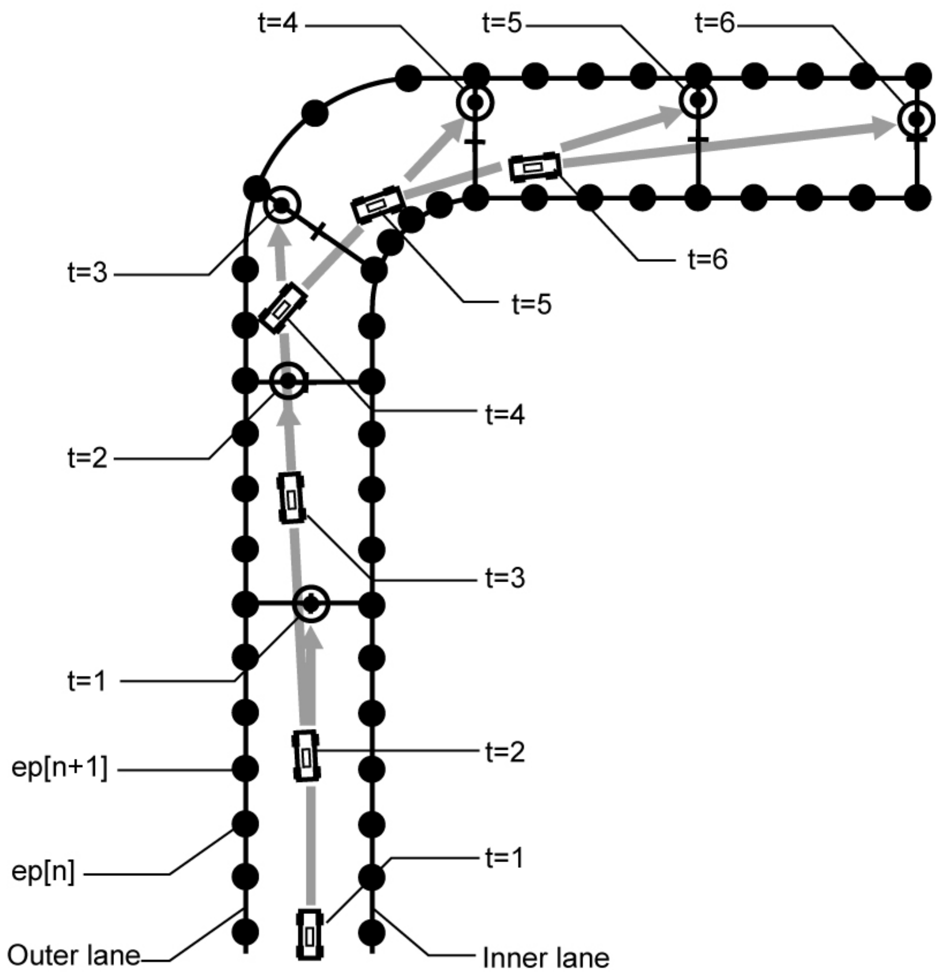


Fig. 21. Offsetting the target to create a larger turning radius for ideal line cornering.

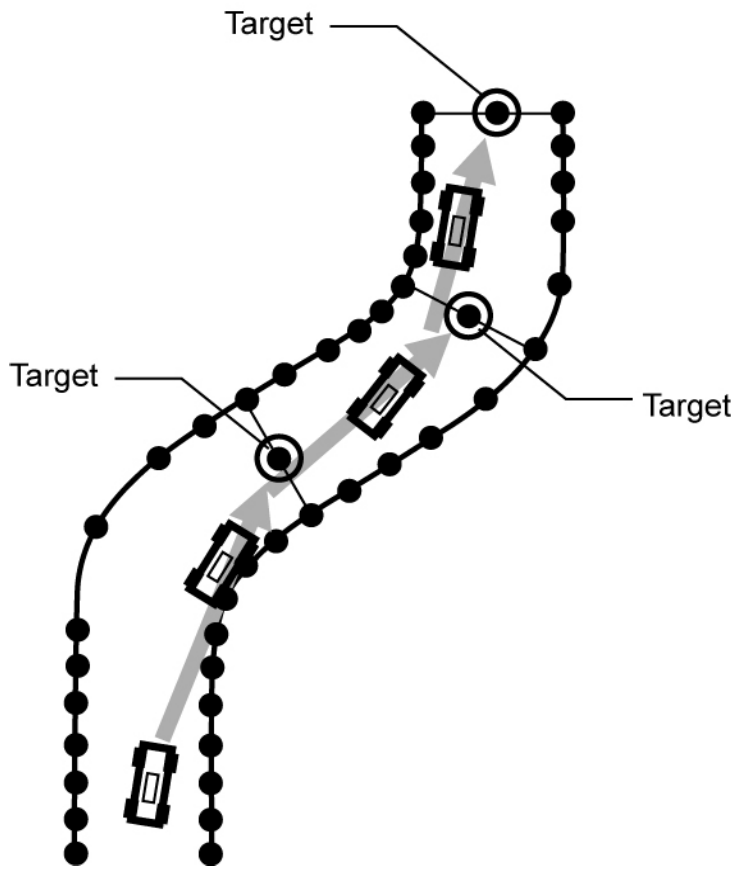


Fig. 22. Cornering on curved roads with short straightaways.

IV.2.2.5. Collision Avoidance

Each car evaluates the distance and position of its opponents to avoid collision (see Figure 23). They will react by braking, passing, or staying the current path. This is determined by evaluating if the opponent is in the “steer left” region, “steer right” region, or “brake” region of the current car (see Figure 24). If the opponent does not fall into any region, the current car will resume its driving path. Each car’s field of view changes with the velocity vector (see Figure 25).

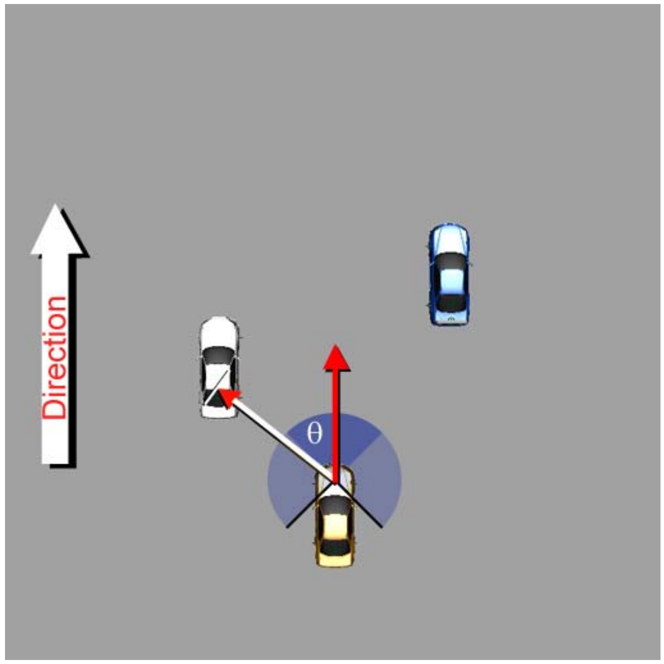


Fig. 23. Checking distances with other cars.

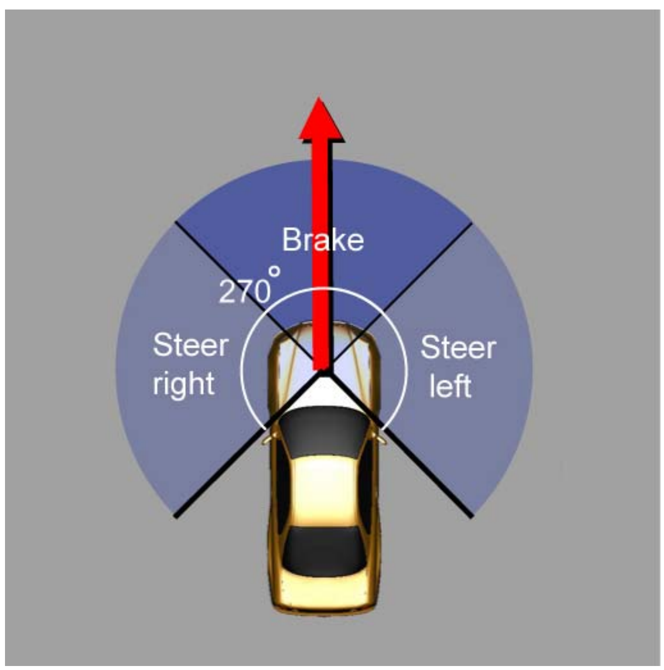


Fig. 24. Field of view and velocity vector.

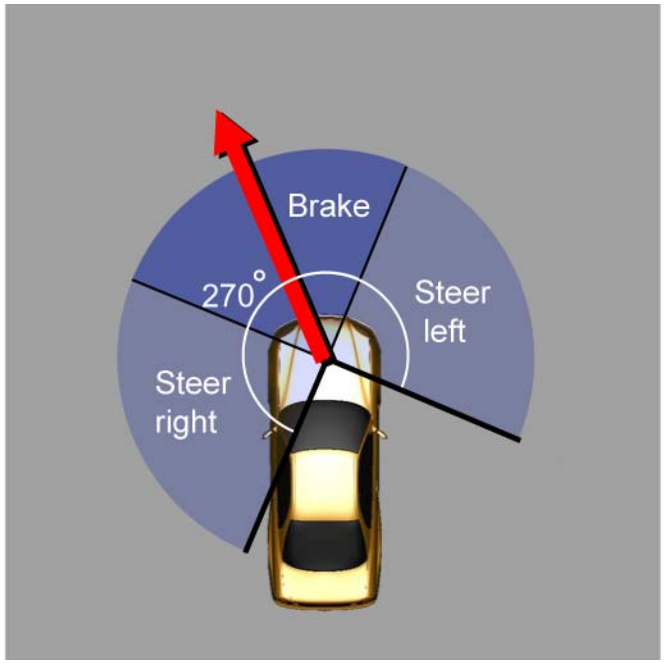


Fig. 25. Angle of view changes with velocity vector.

IV.3. Car Dynamic Attributes

IV.3.1. Realistic Velocity

The car's velocity and acceleration is calibrated to look realistic under all racing conditions. The many combinations of the different performance attributes are scaled to produce a variety of overall performance capabilities that range from a daily commuter to a super car.

IV.3.2. Pitch and Body Roll

When the car is under braking or acceleration, rotation about the z-axis of the car body occurs. Rotation of zero means there is constant velocity. Positive rotation means there is acceleration (Figure 26). Negative rotation means there is braking (see Figure 27). The degree of rotation in either direction is proportional to the rate of change in velocity.

The rotation about the x-axis of the car body corresponds to body roll (see Figure 28). The quantity and direction of body roll is proportional to the velocity of the car and the degree of curvature of the track.

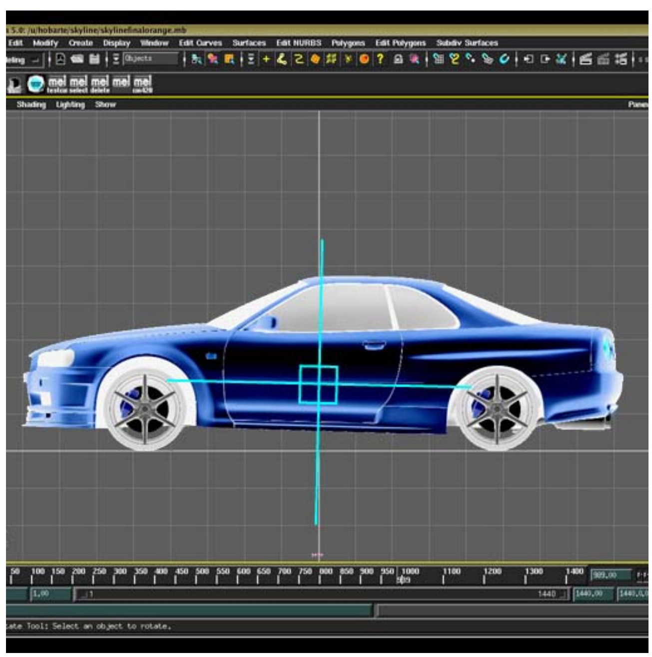


Fig. 26. Back end squat under acceleration.

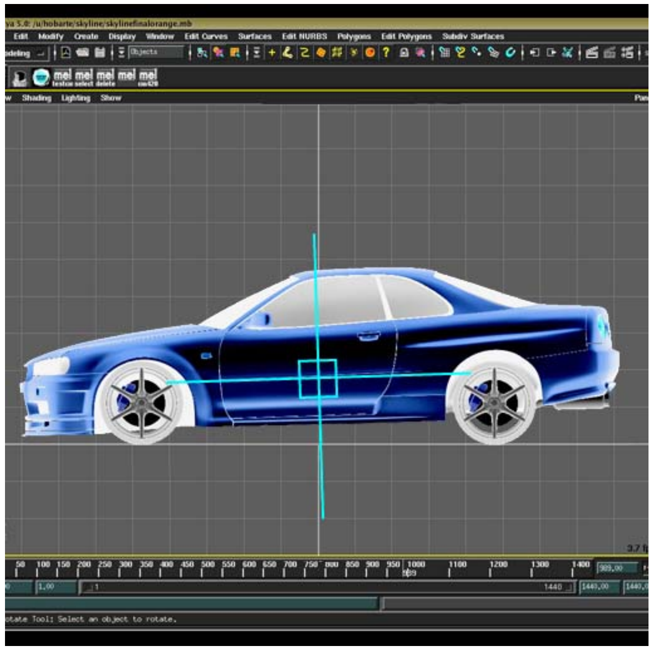


Fig. 27. Front end nose dive under braking.

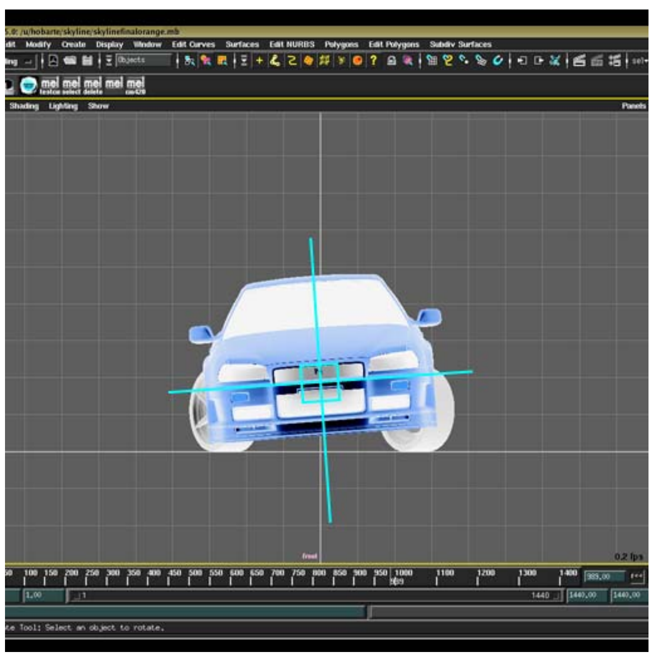


Fig. 28. Body roll during cornering.

IV.4. Simulation Setup

IV.4.1. Track Setup

Track setup begins by drawing a center curve for the racetrack to indicate the path and the direction of the cars (counter-clockwise). The MAYA "rebuild curve" command is then used to specify the number of edit points along this center curve. The edit points represent the evenly spaced critical target locations around the track. The more edit points on the curve, the smaller the spans between target locations. This ensures a smoother, more seamless movement of the target, which in turn affects the movement of the car. The rule of thumb is to have the maximum span about 1/6 of the car's length. If spans are too large, the movement of the target will be unnatural creating abrupt transitions from one location to the next.

To define track boundaries, we use the *MAYA* "offset" command to create two new curves. Each offset from the center by one-half of the track width. This produces two curves, one that defines the outer boundary of the track (outer curve) and the other that defines the inner boundary of the track (inner curve). Edit point index on both curves must increase in the same direction. As we focus our attention on the outer curve and inner curve, the center curve is no longer used and can be deleted.

IV.4.2. Car Setup

A car model must be manually set up for the script to understand. This involves creating and naming key nodes, and creating a hierarchy for the car geometry (Figure 29).

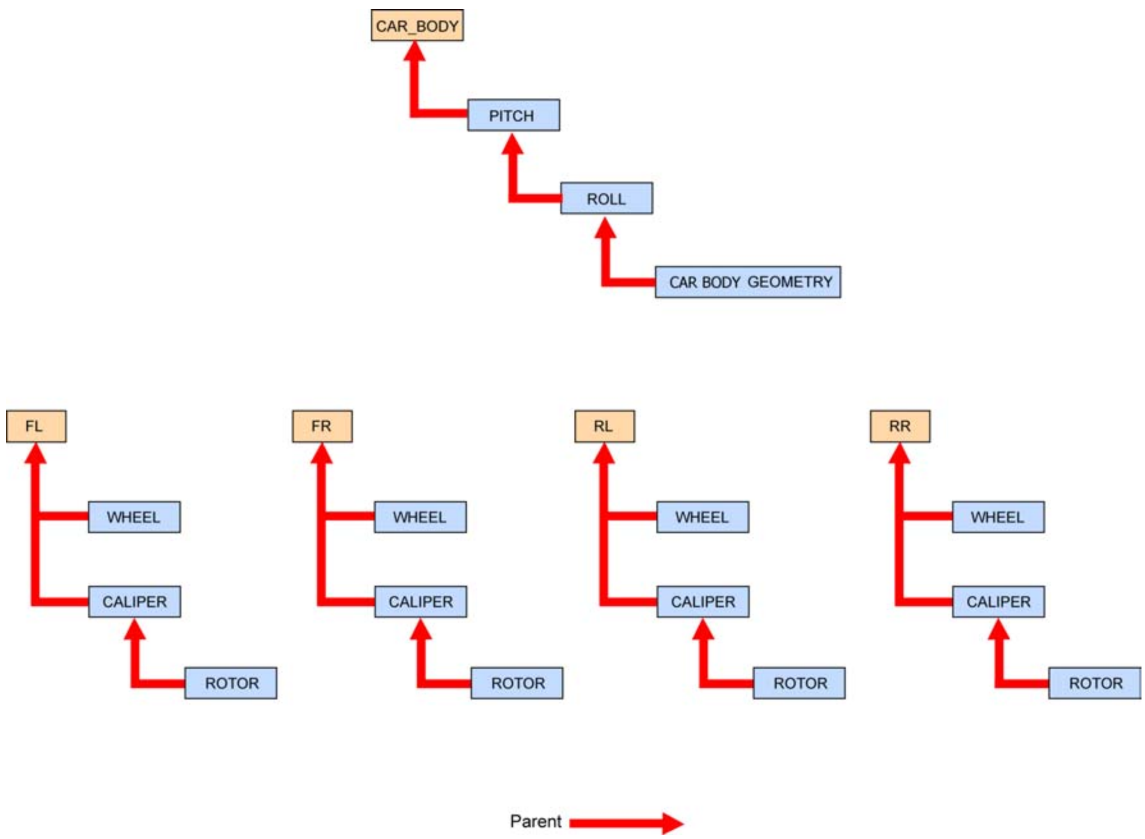


Fig. 29. Car model setup.

The body of a car and the wheels are treated as five separate entities that move independently to allow realistic movement during racing. Body orientation of the car depends on the elevation of the individual wheels. For example, if the front wheels are higher in elevation than compared to the rear wheels, the car must be on a slope and the orientation of the body must follow. The wheels remain in contact with the pavement at all times even when the wheels encounter undulations and bumps in the track.

Pivot point of the “PITCH” and “ROLL” node is placed near the car center between the wheels and at a height slightly above the rolling axis of the wheels (see Figure 30 and Figure 31). The height location is an approximation as the suspension system on each car is at different heights. The “ROLL” node rotates about the x axis while the “PITCH” node rotates about the z axis.

It is important that the caliper and brake rotors turn with the front wheels about the y-axis during steering but not with the wheels about the z-axis during rolling (see Figure 32).

The five main nodes are “CAR_BODY”, which includes all the geometry that make up the body, and “FL”, “FR”, “RL”, and “RR”, which represents front left, front right, rear left, and rear right wheels, respectively (see Figure 33). These five nodes along with the “PITCH” and “ROLL” nodes are then used by the script to initialize the simulation.

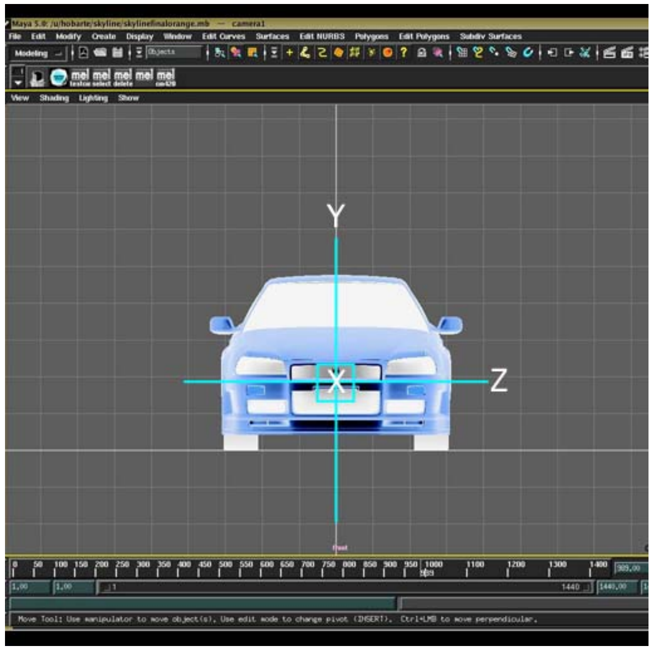


Fig. 30. Pivot point for body roll.

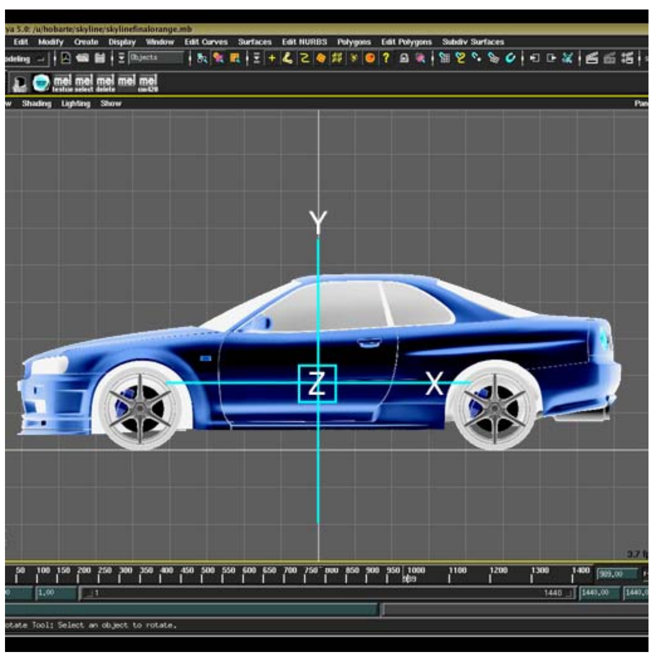


Fig. 31. Pivot point for body pitch.

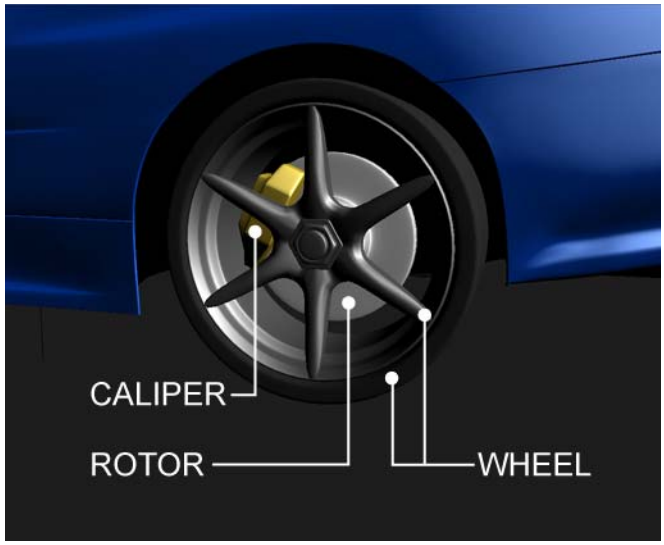


Fig. 32. Example wheel, caliper, and brake rotor.

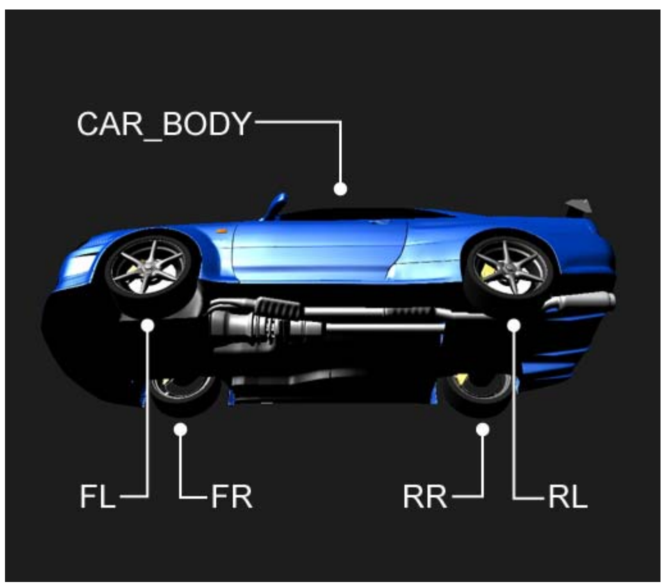


Fig. 33. Main nodes for the car.

IV.4.3. Scripted Car Setup

The car setup script adds more key nodes and creates a more sophisticated hierarchy of the car geometry (see Figure 34). This is generated automatically when the script is executed.

The car reacts to the undulations of the track by a scheme that connects the car body to the wheels. The vertical position of the car body is determined by the average vertical position of all four wheels. The "F MID" and "R MID" locator nodes are "point constrained" to their respective front and rear wheels (see Figure 35). "Point constraint" is a *MAYA* tool that constrains the location of one object to one or more other objects. If the object is constrained to more than one object, a weighted average is used. These locators serve as connection points between the left and right wheels (one for the front and one for the rear wheels) with locations midway between wheel pairs. If the front left wheel is at a higher elevation than the front right wheel, the "F MID" node would be at the average height of the two wheels. A "MID" node then connects the "F MID" and "R MID" nodes together, also with a weighted average. The "MID" node is the "center" point of the car body. The "MID" node only determines the height of the car body. Finally, the nodes are parented under the "CAR_ALL" node that controls all the transformations of the car (see Figure 36). A scheme for determining body rotation is needed.

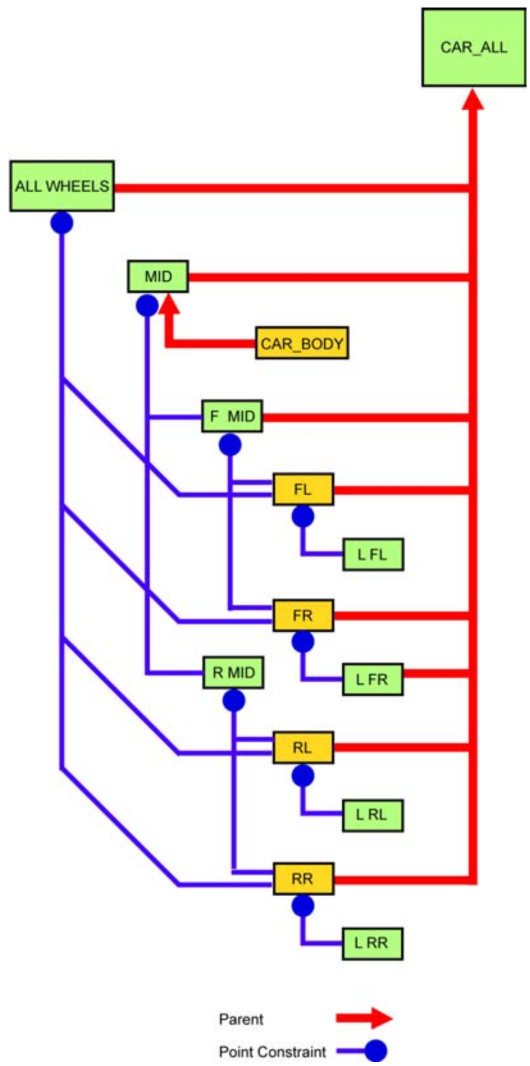


Fig. 34. Diagram of a scripted car rig.

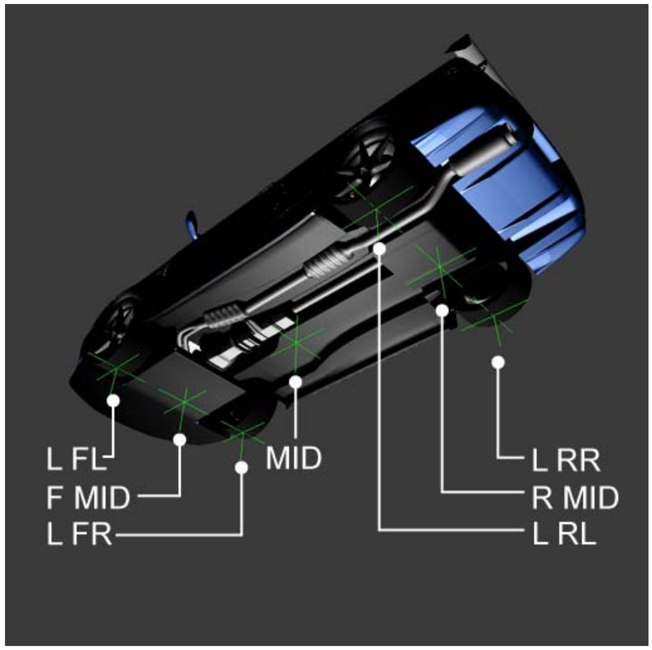


Fig. 35. Locators used to determine body and wheel orientation over uneven surfaces.

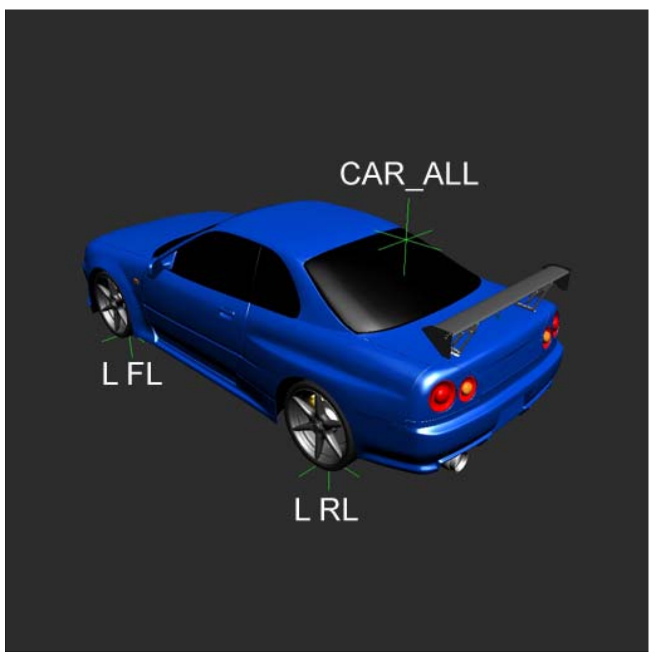


Fig. 36. “CAR_ALL” node controls all the translations and rotations of the car.

As mentioned earlier, the wheels remain in contact with the pavement. The car body reacts to changes in individual wheel height. This is done by using the *MAYA* “geometry constraint” and “normal constraint” tool to keep the “FL”, “FR”, “RL”, and “RR” nodes on the track. It is also at these points (also known as the contact patches) where the “L FL”, “L RL”, “L FR” and “L RR” locator nodes are placed. These locators return the location of all four wheels in world space.

To compute whether the front of the car pitches up or down (rotating on the z axis), we compute the rotation angle of the left set of wheels (see Figure 37) and the angle of rotation for the right set of wheels (see Figure 38) using the equations:

$$\theta_{\text{Left}} = \text{arctangent}\left((\text{L FL.Y} - \text{L RL.Y})/d\right),$$

and

$$\theta_{\text{Right}} = \text{arctangent}\left((\text{L FR.Y} - \text{L RR.Y})/d\right),$$

where θ_{Left} is the angle of rotation of the left set of wheels and θ_{Right} is for the right set of wheels, *.Y is the vertical component of the locators, and d is the wheelbase (distance between the front and back wheels). We then use:

$$\theta_Z = (\theta_{\text{Left}} + \theta_{\text{Right}})/2$$

to compute the overall car rotation about the z axis.

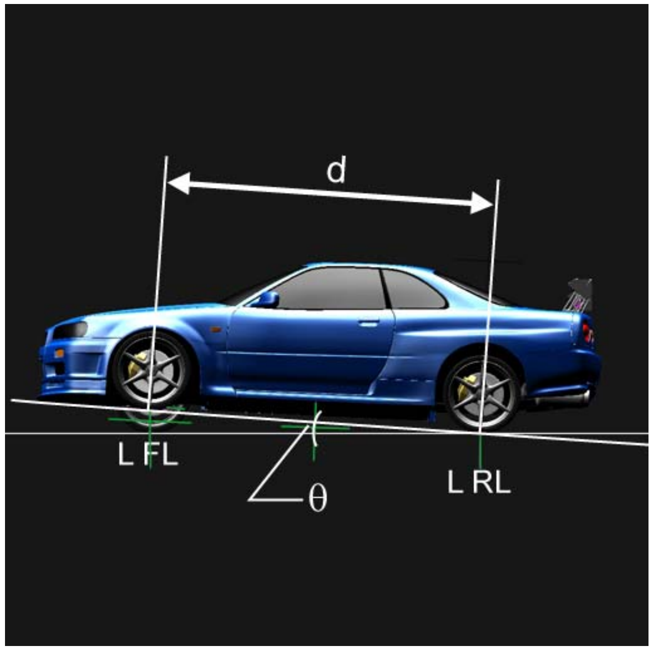


Fig. 37. Left side rotation.

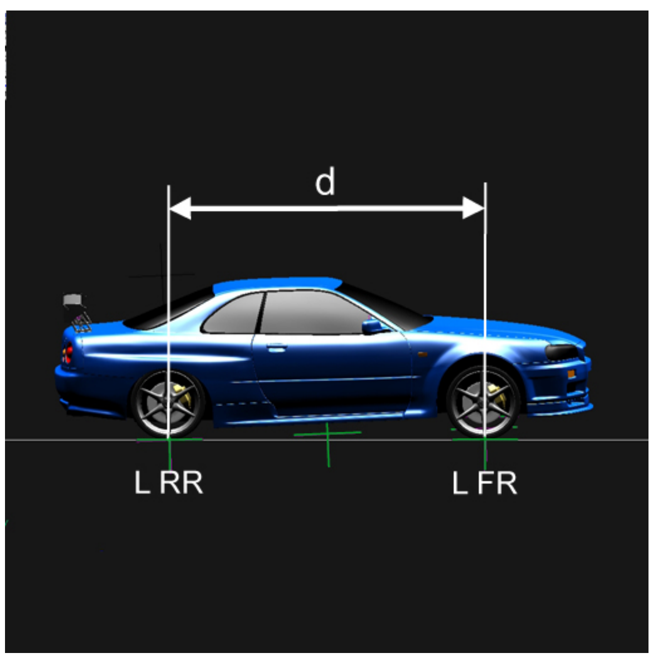


Fig. 38. Right side rotation.

To compute car roll rotation about the x axis, we compute the rotation angle of the front set of wheels (see Figure 39) and the angle of rotation of the rear set of wheels (see Figure 40) using the equations:

$$\theta_{Front} = \text{arctangent} ((L\ FL.Y - L\ FR.Y)/ w_1)$$

and

$$\theta_{Rear} = \text{arctangent} (L\ RL.Y - L\ RR.Y)/ w_2)$$

where θ_{Front} is the angle of rotation of the front set of wheels and θ_{Rear} is for the rear set of wheels, *.Y is the vertical component of the locators, w_1 is the front track width (distance between the front wheels), and w_2 is the rear track width (distance between the rear wheels). We then use:

$$\theta_X = (\theta_{Front} + \theta_{Rear})/2$$

to compute the overall car rotation about the x axis.

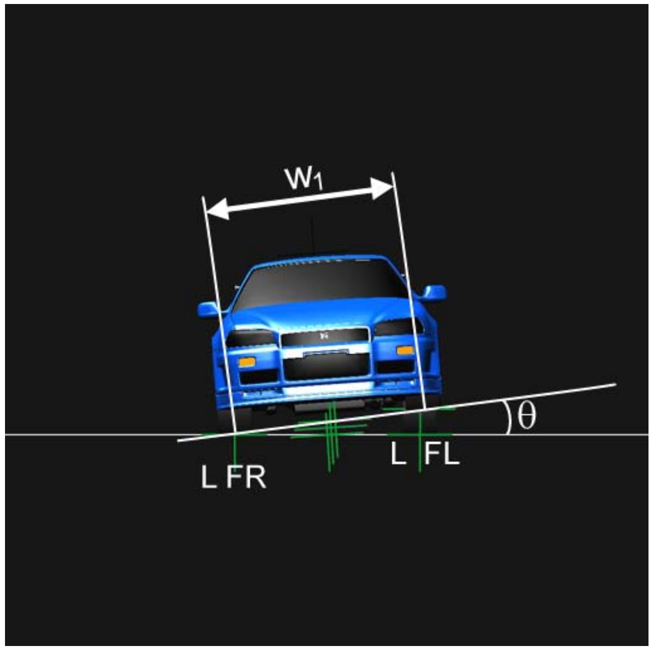


Fig. 39. Front end rotation.

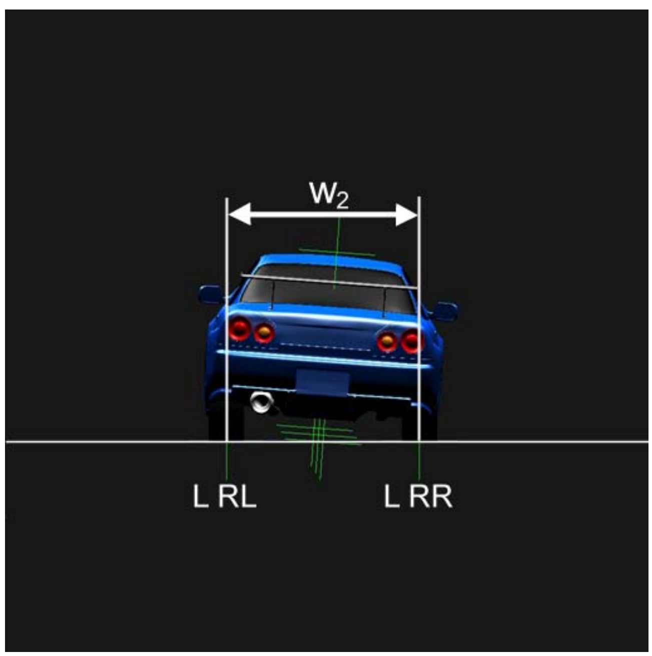


Fig. 40. Back end rotation.

IV.5. Executing the Simulation

Simulation begins by selecting (in this order) the inner curve, the outer curve, and finally the 3D surface, which the cars will run on. The next step is executing the simulation script within *MAYA* script editor (Figure 41).

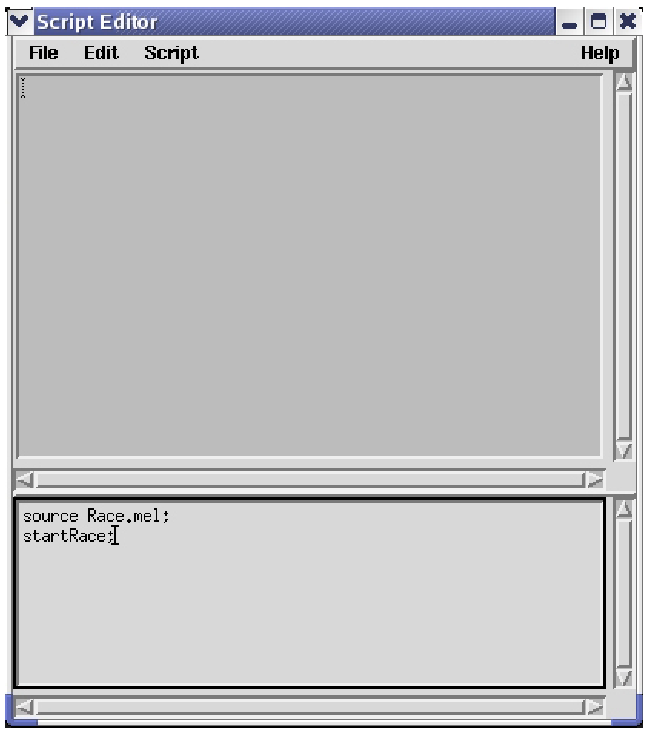


Fig. 41. *MAYA* script editor.

The script then launches a graphical interface requesting the number of cars from a range minimum of 1 to a maximum of 10 (Figure 42). Once the number of cars is chosen, the following window "Select Cars" (see Figure 43) allows the performance attributes of each car to be specified, and to choose the desired car model via the file browser (see Figure 44).

The performance attribute values range from 1 to 10. The combination of different settings changes the performance of a car. The cars can be set to perform like exotic super cars or toned down to perform like daily commuters.

Finally, once all the settings are made, the "Start!" button will execute the script and begins the simulation (see Figure 45).

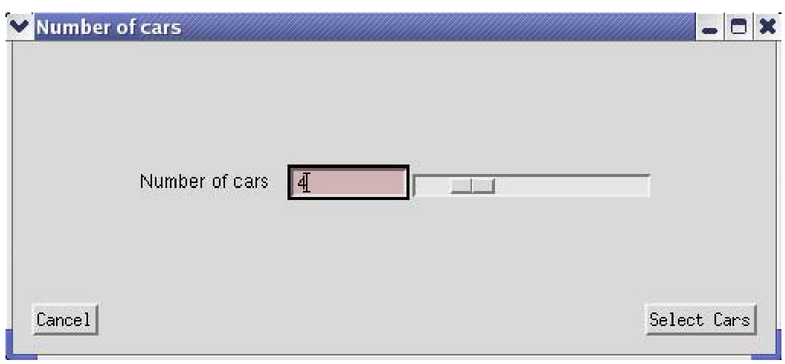


Fig. 42. Interface for controlling the number of cars.

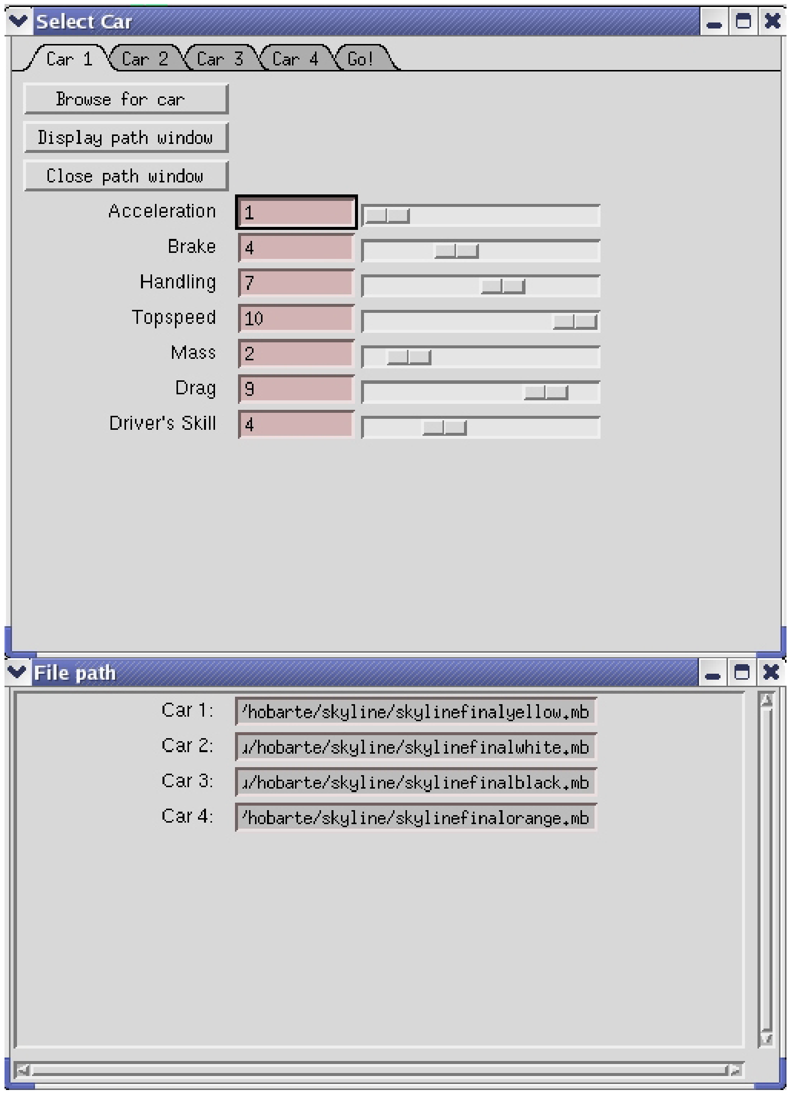


Fig. 43. Interface for car selection and individual attribute settings.

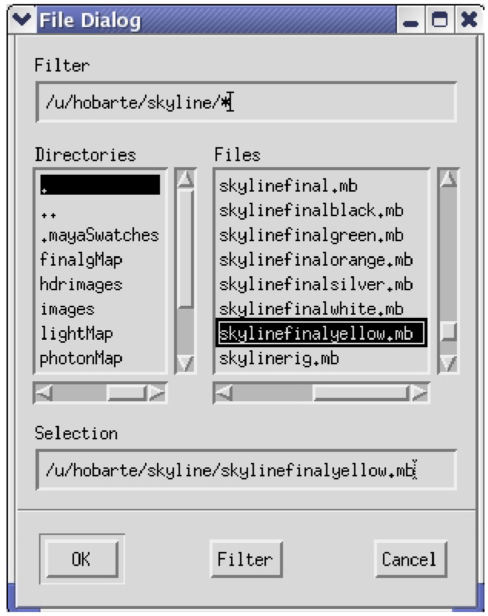


Fig. 44. File browser to select the car models to import.

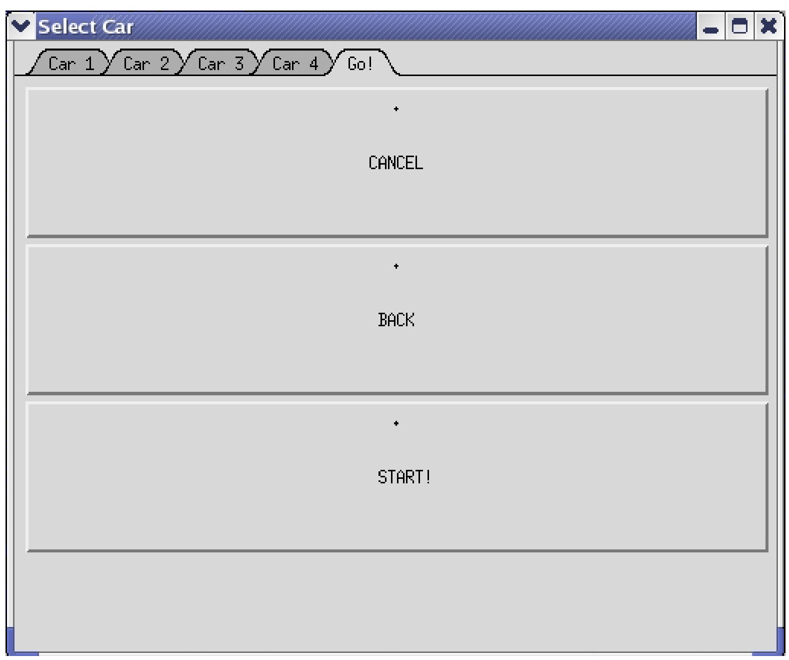


Fig. 45. Go tab featuring final options before executing the script.

CHAPTER V

RESULTS

This simulation system is capable of producing animations of “realistic” car races. In this section, a few example screen shots and image sequences are shown that exhibit racing strategies, racecar dynamics, and driving behaviors. Several racing circuits (modeled after real racetracks found in Japan) were used to test this simulation. These particular racing circuits provide a variety of curves and straightaways that occur in most circuit racing. The cars and racetracks are modeled using the nurbs modeling technique. The textures of the track are painted using the artisan tool in *MAYA* while the environment consists of a sphere with a sky texture mapped on to it.

V.1. Simulation

V.1.1. Track Following

The implemented path following technique keeps the cars within the boundaries of the track. The cars brake, accelerate, and turn smoothly as they follow the targets. Under some circumstances, the cars drive on the shoulder for a short period of time. This occurs during sharp turns as the cars try to maintain straightaway speed. Otherwise, the cars stay on the track. If an off-center driving line was taken (driving toward the left or right boundaries of the track), the cars center themselves back towards the middle of the track when driving on a straightaway.

V.1.2. Real World Velocity

On the highest performance settings (a setting of 10 for acceleration, braking, handling, top speed, driver's skill, and a setting of 1 for mass and drag), a car ran from 0–60 miles per hour in 3.5 seconds whereas on the slowest settings, a car will take 15 seconds to do the same. The fastest car driven by the most skillful driver finished a 2.46 mile lap 2 ½ times faster than the least skillful driver with the slowest car settings. The highest top speed recorded was 180 miles per hour on the straightaways whereas some cars topped out at 70 miles per hour. During cornering, some cars slowed down to 40 miles per hour while some went through at 80 miles per hour.

V.1.3. Cornering and Steering

The look-ahead distance allows each car to anticipate track conditions. If a curve connects to a straightaway, the car will know ahead of time to take the ideal line. The car begins on the inner boundary of the track but ends up closer to the outer boundary of the track by the time it hits the turn-in point (see Figure 46). This is to prepare for a late turn-in and a late apex. Full throttle is applied right after the apex for a quick exit speed while maximizing the use of track width.

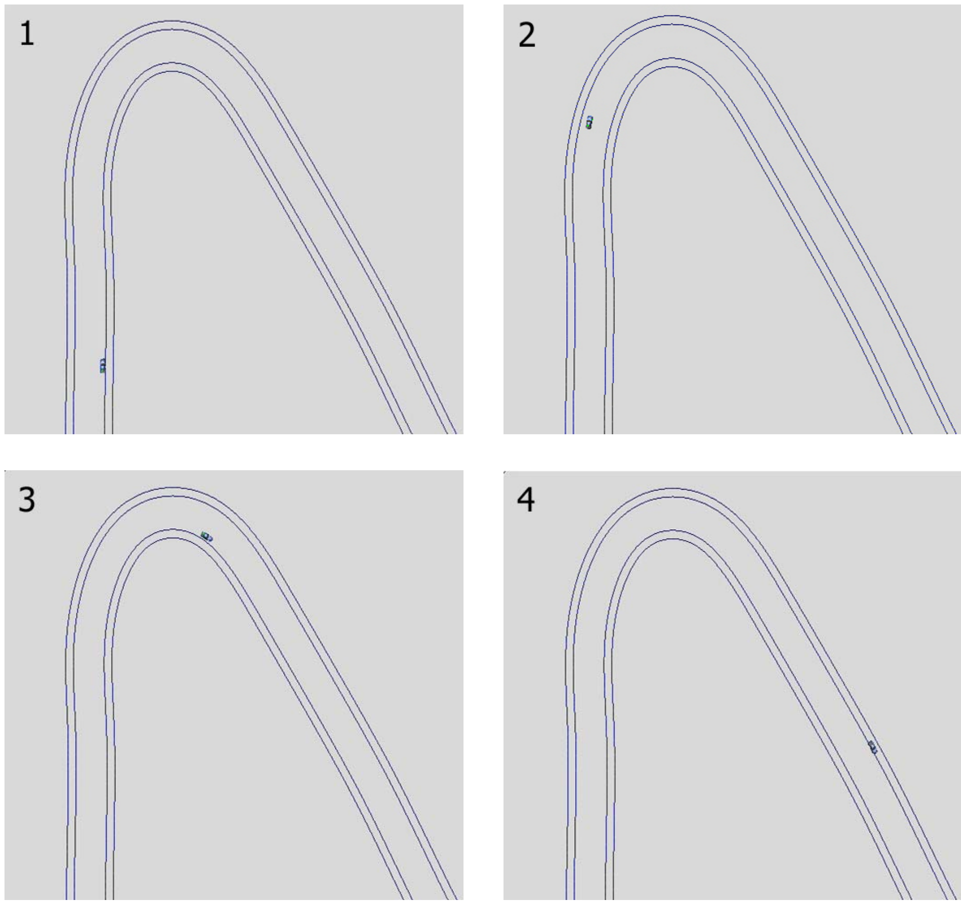


Fig. 46. Taking the ideal line through a curve that connects to a straightaway.

For consecutive curves, the car will deviate from the geometric line. The car maximizes the turning radius by staying closer to the outer boundary during the full turn (see Figure 47). If the last curve connects to a long straightaway, the car will position itself to take the ideal line. As each corner is different, the car automatically blends between the ideal line and the geometric line as needed.

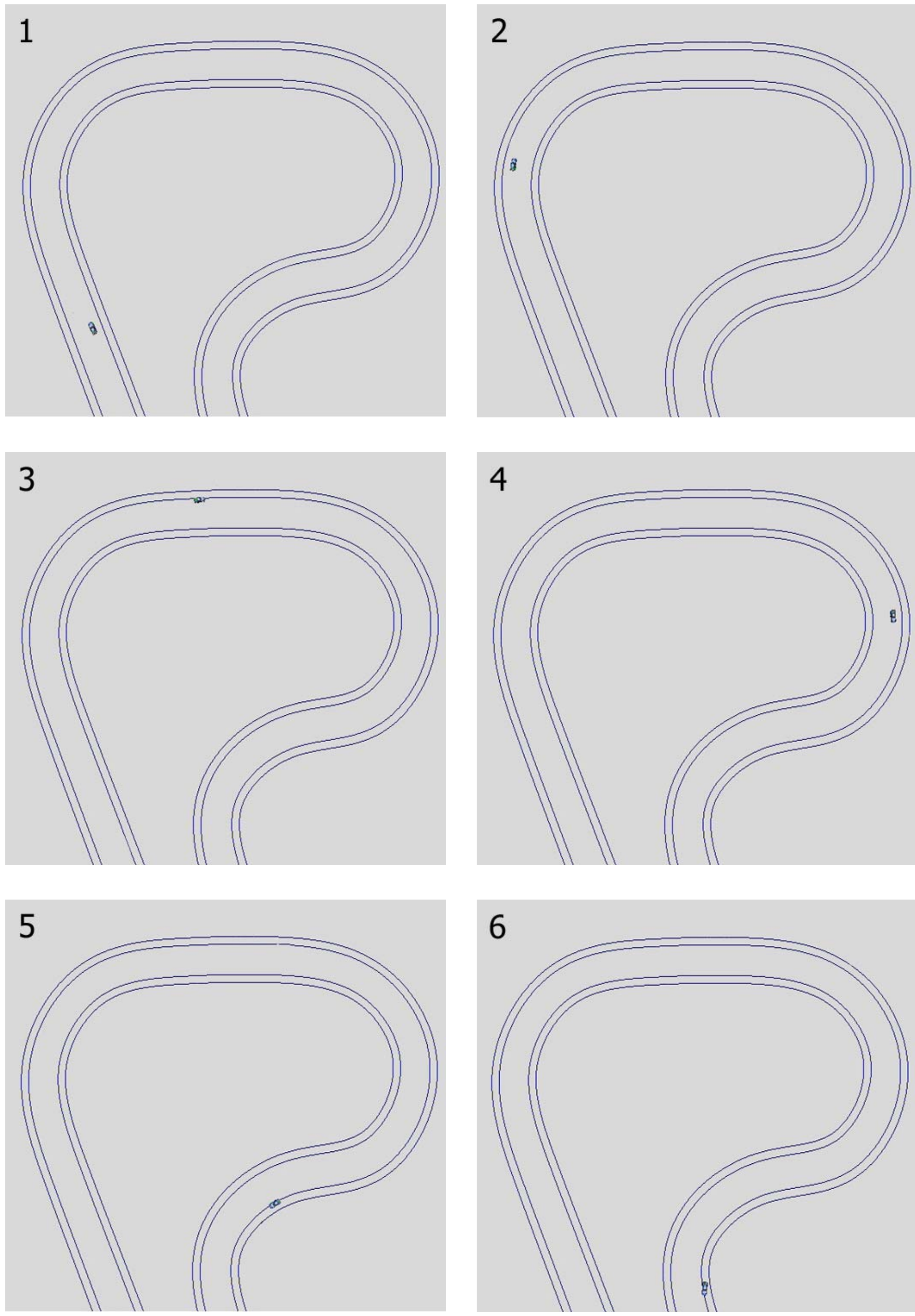


Fig. 47. Taking the geometric line on two consecutive turns.

The same driving line is used when taking S-curves (Figure 48). Entrance corner speed is maximized while taking the largest inscribed turning radius.

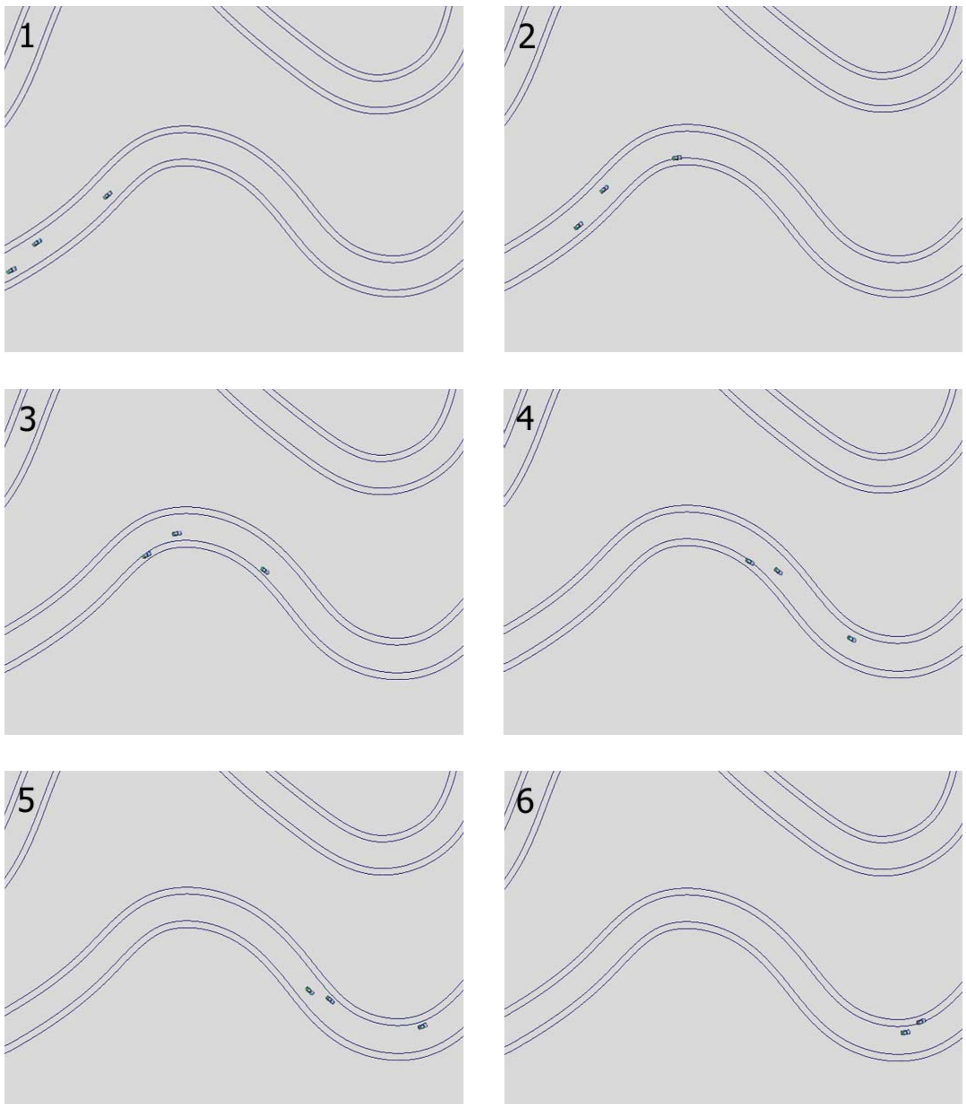


Fig. 48. Taking the geometric line through an S-curve.

V.1.4. Collision Avoidance

Avoiding collision provides fluid motion among the cars that are near each other. Aggressive drivers with faster cars tend to pass slower opponents easily. The less aggressive drivers attempt to pass, yet get stuck behind the opponent on both during cornering and accelerating on straightaways. This is because both cars are taking the optimum driving path, and neither is willing to take an alternate path, as is true in the real world.

V.2. Dynamic Attributes

The realism of this simulation is also due to having visual cues that the cars are reacting to the behaviors of the driver. During a race, each car exhibits the visual cues such as squat, dive, and roll associated with acceleration (Figure 49), braking (see Figure 50), and cornering (see Figure 51, Figure 52, and Figure 53).

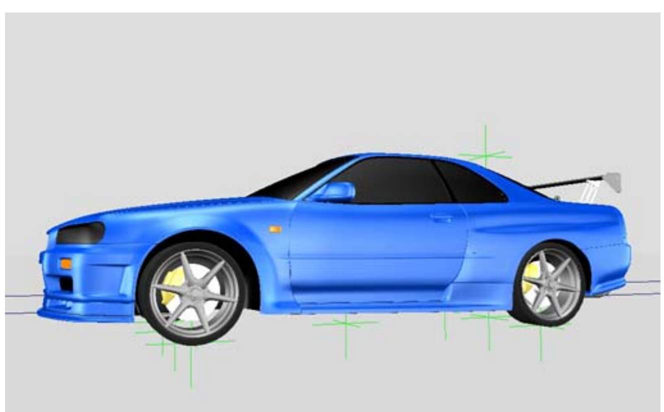


Fig. 49. Straight-line acceleration.

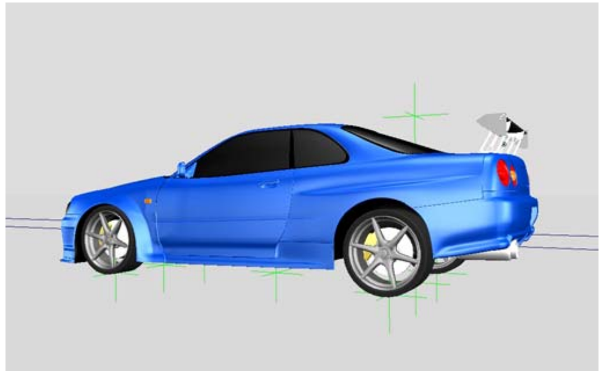


Fig. 50. Straight-line braking.

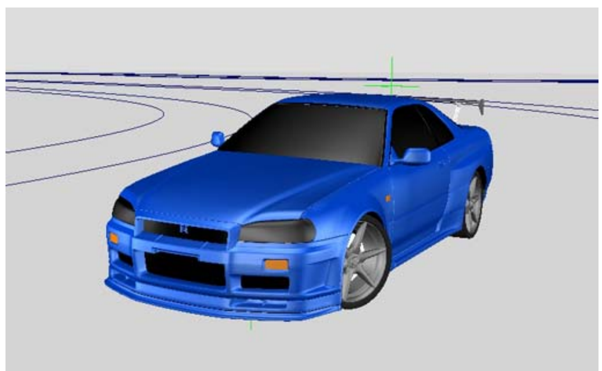


Fig. 51. Front view of a car cornering and braking.

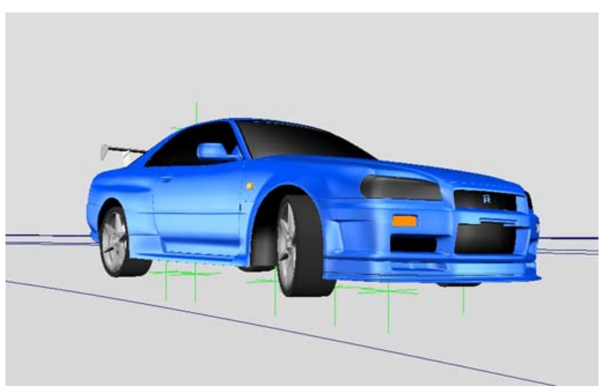


Fig. 52. Front view of a car cornering.

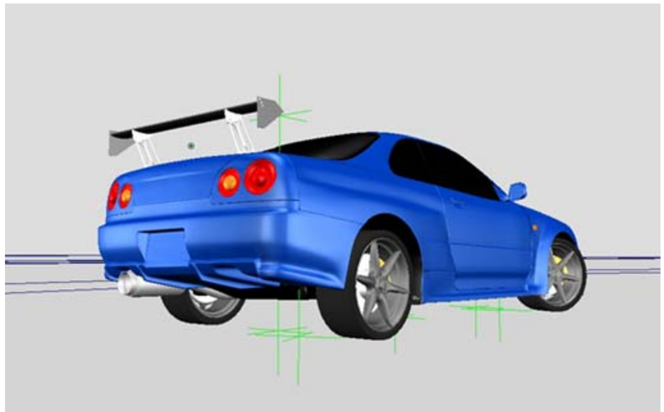


Fig. 53. Rear view of a car cornering and braking.

When driven over uneven surfaces, the body motion of the car reacts to the elevation changes of each individual wheel (Figure 54). Each individual wheel remains in contact with the pavement at all times (see Figure 55).

Fig. 54. Driving over an uneven surface.

Fig. 55. Close-up of the right rear wheel over an uneven surface.

As the car is driven over hills and slopes, the position and rotation of each wheel responds to the surface (Figure 56). This in turn affects the overall position and rotation of the car body (see Figure 57 and Figure 58).

Fig. 56. Aerial view of a car driving on a slope.

Fig. 57. Front view of a car driving on a slope.

Fig. 58. Rear view of a car driving on a slope.

CHAPTER VI

CONCLUSION AND FUTURE WORK

VI.1. Conclusion

In this research, we have addressed the development of a simulation system that produces realistic car racing behavior. The system created meets the initial objectives and has been used to create successful racing animation examples. Since this project is developed in *MAYA*, the possible final rendering styles are endless. This work can be used for live action effects production or for traditional computer animation. In addition, the procedures written in the script do not restrict usage to one type of car model or track model. The methods of this system work much like a video game but without constant control input required from the user.

VI.2. Future Work

Although this system models realistic car racing behavior, it still does not model all aspects of car racing behavior and environmental conditions.

Unlike circuit racing, which is usually held during daytime, other races such as the 24 hours of Le Mans is partially held at night. As daylight diminishes, drivers only see what the headlights illuminate. Driving becomes much more difficult and the driving style must adapt. Another condition that race drivers often face is driving in the rain. Not only does rain cut down on visibility, it also diminishes tire traction, which affects overall driving behavior. Future work could address these additional racing conditions.

Path finding is an interesting aspect of racing. Not all races are like circuit races, which are held on a closed track (where drivers know the course). Other races such as rally racing relies on a driver's instinct and intelligence to find paths to their destination. This is also true for races held in the city (illegal street racing and car chase scenes). The simulation of path finding is more complex because there are many more factors to consider. Rally racing involves driving on unpredictable terrain such as dirt, sand, water, and even snow whereas street racing involves the many other activities that go on in a city (pedestrians, road hazards, and other vehicles). As conditions change by the second, so do the many behaviors of the driver. Path finding is another challenge that can be addressed in future work.

Finally, the use of a more accurate physical simulation of the dynamics of a car will achieve even greater realism. This should accurately model torque and horsepower output, coefficient of tire adhesion, and aerodynamics, which will affect car's overall performance. The car should also be able to collide with other objects much like a real car would. Although racing involves driving on relatively flat roads, there are conditions where a car is capable of being partially airborne. Accurately modeling a car jump as well as the landing will greatly increase, not just the realism factor, but also the entertainment factor.

REFERENCES

[1] S. Amkraut, M. Girard and G. Karl, "Motion studies for a work in progress entitled *Eurythmy*" in *SIGGRAPH Video Review*, Issue 21 (second item, time code 3:58 to 7:35), 1985, produced at the Computer Graphics Research Group, Ohio State University, Columbus Ohio.

[2] R. Bentley, *Speed Secrets: Professional Race Driving Techniques*. MotorBooks International. 1998.

[3] M. Calderon, "Avoiding character collisions in games", M.S. thesis, Texas A&M University, 1999.

[4] C.S. Carpenter, B.W. Seales, C. Jaynes and R. Stevens, "Automated basis-view and match-point selection for the ArchVision RPC image-based model", *Proceedings of the Ninth ACM International Conference on Multimedia.* pages 577-581, Ottowa, Canada, October 2001.

[5] P. Frere and P. Hill, *Sports Car and Competition Driving.* Bentley Publishers, 2003.

[6] D. Koeppel, "Massive Attack", *Popular Science*, November 2002. http://www.popsci.com/popsci/science/article/0,12543,390918,00.html accessed: March 2004.

[7] P. Krzeminiski, "Softimage behavioral procedural crowd animation", *RenderNode,* September 2003. http://www.rendernode.com/articles.php?articleId=66 accessed: March 2004.

[8] J. Kundert-Gibbs and P. Lee, *Mastering Maya 3*. Cheryl Applewood, 2001.

[9] S.R. Musse and D. Thalmann, "From one virtual actor to virtual crowds: Requirements and Constraints", *Proceedings of the Fourth International Conference on Autonomous Agents*. pages 52-53, Barcelona, Spain, 2000.

[10] S. Rabin, *AI Game Programming.* Charles River Media. 2002.

[11] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model", *Computer Graphics*, Vol. 21, No. 4, pp.25-34, 1987.

[12] C. W. Reynolds, "Interaction with groups of autonomous characters*", Game Developers Conference Proceedings,* pages 449-460, San Francisco, CA, 2000.

[13] C. W. Reynolds, "Steering behaviors for autonomous characters", *Game Developers Conference Proceedings*, pages 763-782, San Jose, CA, 1999. http://www.red3d.com/cwr/steer/ accessed: October, 2003.

[14] Rich Photo Realistic Content Cars, Official website, http://www.archvision.com accessed: March, 2004.

[15] M. Wilkins and C. Kazmier, *MEL Scripting for Maya Animators.* San Francisco: Morgan Kaufmann Publishers, 2002.

[16] A. Witkin and D. Baraff, "Physically based modeling", *SIGGRAPH '99 Course Notes, Course 36*, SIGGRAPH '99, August 1999.

VITA

Hobart Chan
17234 Pastoria Dr.
Houston, Tx 77083
hobarte@hotmail.com

Education

M.S. in visualization sciences Texas A&M University, August 2004
B.E.D. in environmental design Texas A&M University, May 2001

Employment

Intern Architect, Gossen Livingston Associates
May 2001 – August 2001
May 2002 – June 2002
Intern Architect, Merriman Holt Architects
May 2000 – August 2000
December 2000